Vrije Universiteit Brussel

FACULTY OF SCIENCE
DEPARTMENT OF COMPUTER SCIENCE

# A Framework for Real-Time Emotion Recognition with an Online EEG-based BCI during Content Browsing

Graduation thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Applied Sciences and Engineering: Computer Science

Sam Nielandt

| | |
|---|---|
| Promoter: | Prof. Dr. Beat Signer |
| Advisors: | Sandra Trullemans |
| | Brecht De Rooms |

Academic year 2013-2014

Vrije Universiteit Brussel

FACULTEIT WETENSCHAPPEN
VAKGROEP COMPUTERWETENSCHAPPEN

# A Framework for Real-Time Emotion Recognition with an Online EEG-based BCI during Content Browsing

Afstudeer eindwerk ingediend in gedeeltelijke vervulling van de eisen voor het behalen van de graad
Master of Science in de Ingenieurswetenschappen: Computerwetenschappen

Sam Nielandt

| | |
|---|---|
| Promoter: | Prof. Dr. Beat Signer |
| Begeleiders: | Sandra Trullemans |
| | Brecht De Rooms |

Academisch jaar 2013-2014

# Abstract

Due to the technical evolution of brain controllers, innovative Brain-Computer Interfaces (BCI) are developed in order to detect mental activity, emotions or to aid people with a paralysis. In human-computer interaction, BCIs have the potential to provide users information such as their emotion state which can be used to personalise applications. The research field of BCI spans multiple domains like neuroscience, psychology,signal processing and machine learning. Furthermore, different technologies can be used to acquire brain signals. In this thesis, we have opted for an Electroencephalography (EEG) interface with only four sensors to increase user-friendliness. A major issue with existing software solutions is the lack of support for non-expert developers in BCI and often they are limited to specific domains. Therefore, we present an extensible framework that allows applications to use real-time emotion recognition through the use of an EEG-based BCI. Our framework provides a solution for the development of BCIs by non-expert developers. Simultaneously, it still provides expert developers the flexibility of using an own design for the BCI embedded in our framework. Furthermore,our framework is generic in that it can be used to recognise emotions evoked by different types of stimuli such as visual, auditory and audio-visual content. In addition, our framework can be used with different EEG capture devices so developers and users are not forced to use a specific device.The BCI embedded in our framework was specifically designed to work with only four electrodes to increase user-friendliness. Also, we aimed for a BCI that classifies emotions in real-time to increase development possibilities for applications that use our framework.

Due to the multidisciplinary aspect of BCI, we first explored relevant research from different research areas. In order to understand where brain signals come from and how EEG-signals are generated, we investigated the working of the brain as well as the functional areas of the brain. This neuroscientific insight enabled us to get a better understanding of how the location source of brain signals can be used to achieve more accurate classification results. Research on signal processing techniques for preprocessing and feature extraction on EEG-signals and machine learning techniques for classifying EEG-signals based upon extracted features provided us with an understanding of how BCIs are designed and which different techniques can be used towards detecting specific mental activity. Since our goal was to design a framework for emotion recognition, we also performed research work theclassification of emotions in psychology as well as how this representation can be translated into a design concept for a BCI.

The framework itself consists of an emotion recognising BCI surrounded by a communication layer written in Java. The communication layer is able to start up the BCI and to send identifiers to the BCI through a shared buffer to label the incoming EEG signal for the purpose of training the BCI as well as terminating the BCI. Furthermore, the communication layer receives classification results from the BCI over a Virtual Reality Peripheral Network (VRPN) server. The classification results are then stored in a queue. The communication layers offers easy access to functionality for applications that lie on top of the framework in order to retrieve classification results from the queue, start and stop the BCI and label the incoming EEG signal to train the BCI.

Based on test results, our framework offers an accuracy of 57.1% for recognising 4 emotions—which are happy, anger, calm and sad—while browsing pictures. The test results were obtained by performing a test experiment with an application we developed on top of our framework to recognise emotions while browsing pictures.

# Acknowledgements

I would like to thank my advisors, Brecht De Rooms and Sandra Trullemans, for guiding and helping me in the process of writing this thesis. I also want to thank my promoter, Prof. Dr. Beat Signer, for helping me to improve the quality of my thesis. My thanks goes out to Lien Van Der Biest as well for providing me with books and other reading material on psychology and neuroscience subjects. Also, I would like to thank all the people that helped me out by participating in the test experiment included in this thesis. Finally I would like to thank my father and stepmother for providing and caring for me and for giving me the opportunity to study.

# Contents

# 1 Introduction

## 1.1 Well, It's Not a Mentalist Trick

Throughout the ages, the working of the brain has always fascinated the human kind. In 1920, a German scientist named Hans Berger started exploring ways to measure brain activity by representing brain activity as electric signals. This resulted in 1929 in the first electroencephalography (EEG) recordings of brain activity. Over time, the idea grew to use EEG recordings as a way to communicate brain activity to a device. In 1970, the Defense Advanced Research Projects Agency (DARPA) of the USA started a program to begin the first research on Brain-Computer Interfaces (BCI). BCI are intended to form the link between brain and computer by interpreting signals from the brain. By giving meaning to incoming brain signals, BCI enables the use of brain signals as an input for several applications. In other words, BCI enables users to drive applications by thinking.

Applications range from research on the brain to control of objects in a virtual reality with the mind or machines in the real world. BCIs can be used for instance to control electrical prostheses [25][43]. This enables people who lost a limb to use an electrical prosthesis by simply imagining movements. This concept can be taken even further by using a neurochip to capture brain activity [13]. Another application of a BCI is the control of an electric wheelchair [40]. BCIs can also be used as a way of communication for

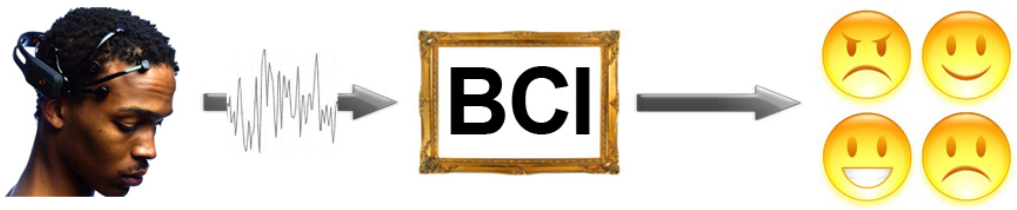


Figure 1.1: Diagram of the emotion recognition of our framework with the embedded BCI

fully-disabled people by controlling a speller [12]. The speller lays out the alphabet in a raster and randomly flashes rows and columns of the raster. The user needs to focus on the character to be spelled. Whenever the row or column flashes that contains the character, the brain responds with a typical signal. The BCI recognises the evoked signal in the brain and can determine the row and column of the character based on this typical signal. For the use of BCI as a means for controlling videogames, an example includes an implementation with the famous game Space Invaders [7]. Also on mobiles device BCIs are applied, for example, with the game BreakOut [33].
In this thesis we explore the world of BCIs and investigate the use of BCIs in the recognition of emotions evoked in a user while browsing content in particular. The goal of this thesis is to provide an extensible framework that can be used to recognise emotions of users in the development of several applications. The idea is to build a framework around an emotion recognising EEG-based BCI in a generic way to enable applications to receive emotional feedback from a user who is browsing through content, as illustrated in Figure 1.1. Through research on the concepts of BCIs and the use of EEG-based BCIs in the recognition of emotions in brain signals, we were able to present a generic, flexible and extensible framework for non-expert and expert developers. As a proof of concept, we built our own application, named Emages, on top of this framework. The Emages application aims to provide users the ability to browse through pictures and automatically label these images according to emotions they felt. Emages also enables users to retrieve these images based on the observed emotions. It should be noted that the framework we propose is intended to be used for the recognition of emotion evoked by different types of content and not only images.

## 1.2 Contributions

BCIs are mostly designed to work with specific applications in a particular domain. This leads to the fact that for each application a new BCI needs to be designed along with a means of communication between the BCI and the application as well as a way to process the results of the BCI. Furthermore, developing BCI-driven applications requires developers to have a solid background in BCI design and techniques. This thesis adresses this problem by introducing an extensible, flexible and generic framework that offers all the functionality that is needed to use an emotion recognition EEG-based BCI without the required knowledge of the design and techniques of the embedded BCI. For expert developers the framework still offers the flexibility to change the design of the BCI to further tweak it. Our framework is generic in that it can be used to develop any type of application that intends to use emotion recognition through an EEG-based BCI and that the BCI embedded in the framework can recognise emotions for any type of stimulation. For example, stimulations of a visual, auditory or audio-visual nature can be used or memories can be used to evoke the emotions to be recognised.
The requirements and challenges we pose to our framework are contributions on itself as well. Firstly, we want our framework to be able to recognise emotions in real-time. EEG-based BCIs that recognise emotions in real-time are rather rare in the literature. Real-time emotion recognition by BCIs offers a wider range of application possibilities. For example, if an application wants to support a user based upon a negative emotion the user is feeling at that moment, a real-time emotion recognition is required. However, the down-side of real-time emotion recognition by BCIs is that some techniques cannot be used to work in real-time which makes it harder to achieve an accurate classification result. Furthermore, we want the BCI embedded in our framework to work with as few sensors as possible. A low number of sensors greatly increases user-friendliness. By reducing the hassle of installing an EEG capture device with a lot of sensors to connect on the head, the use of EEG-based BCI emotion recognition becomes less intrusive. We like to contribute by presenting the implementation of our BCI and our accuracy test results for the purpose of comparison.

## 1.3 Thesis Structure

We begin this thesis with Chapter 2, which deals with some background theory in the field of neuroscience in order to better understand the concepts of a BCI. In Chapter 3 we discuss the concepts and the components of a

BCI as well as different ways to measure brain activity. However, we will focus on the concepts of BCIs that use EEG to measure brain activity. The next chapter is Chapter 4 in which we discuss how emotions are described by theoretical models in the field of psychology and how EEG-based BCIs that use these models can recognise emotions. This brings us to Chapter 5 in which we present an approach to our framework based on the research we carried out and presented in the previous chapters. Next we discuss an implementation that serves as a proof of concept in Chapter 6. Finally we present our test and results in Chapter 7 and draw our conclusions in Chapter 8.

# 2 Neuroscientific Background

This chapter deals with some neuroscientific background theory. Topics contain the architecture of neurons and how they communicate as well as brain areas and their functions. This information helps the understanding of where brain signals are coming from and how they are generated. Furthermore, this chapter helps to clarify the meaning and working of certain concepts in later chapters.

## 2.1 The Neuron

The core components of the brain are neurons and glial cells. Glial cells, also called nueroglia, regulate the internal environment between neurons. They guide the development of neurons and nutrify them. In other words, glial cells act as support cells for neurons. Together with the axons of neurons, glial cells make up what is called the white matter of the brain. The cell bodies of the neurons make up the grey matter of the brain [42]. The human brain is made up of approximately 100 billion to 1 trillion neurons [5]. However, neurons, as well as glial cells, are not only present in the brain. They are found throughout the whole nervous system as well. The reason for this is that neurons are cells that are specialised in sending information to specific target cells. Figure 2.1 shows the diagram of a neuron and we will discuss the parts of the neuron below.

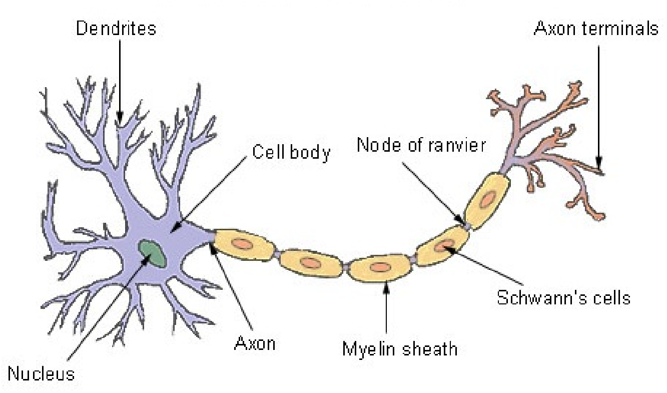


Figure 2.1: A neuron

- **Cell body**
  The body of the neuron, also called soma, contains the nucleus. The nucleus houses most of the neuron's genetic material and is responsible for protein synthesis [5]. It is the control center of the cell.

- **Dendrites**
  The dendrites are cellular extensions connected to the soma of the neuron. They have many branches that each contain many dendrite spines. These dendrite spines are little extensions that contain different types of chemical receptors and are connected to the synapses of other neurons [5]. A dendrite spine can store signals it receives from a synapse. This implies an anatomical substrate for memory storage. Furthermore, dendrite spines can transmit received signals through the dendrites to the body of the neuron. Because of this we can view the dendrites, along with its spines, as the input of a neuron.

- **Axon**
  The axon is a cable-like extension of the body of the cell. The part were the axon is attached to the body of the neuron is called the axon hillock. This part of the axon generates spikes in the cell membrane, called action potentials, that are transmitted through the axon [5]. Besides the fact that the axon is generally more involved in the output of the neuron, it can also receive input from other neurons.

- **Axon terminals**
  The axon terminals contain the synapses of the neuron. Synapses are the contact points of the neuron to the dendrite spines of other neurons to transmit information through the use of neurotransmitters. Neuro-

transmitters are biochemicals and are stored in tiny containers in the axon terminals, called synaptic vesticles. The synaptic vesticles can be triggered to release the neurotransmitter they contain to send their neurotransmitters to dendrite spines of connected neurons. The dendrite spines of the neuron that receives the neurotransmitters contains chemical receptors that bind with the neurotransmitters to exchange the embedded information [5].

We mentioned that neurons are cells that can share information with one another. A neuron can share information by sending electrochemical signals which are triggered by changes in the membrane potential of the neuron. The membrane potential of a neuron is the difference in electrical potential between the interior and the exterior of the neuron. The interior and exterior of a neuron is divided by the cell membrane of the neuron, which acts as an insulation and diffusion layer for ions in both the interior and exterior of the neuron. The cell membrane is embedded with proteins. Some of these proteins act as ion channels or ion pumps. Ion channels can be switched between an open or closed state to regulate the flow of ions by allowing a specific type of ions to pass through the neuron's membrane to enter or leave the membrane. Ion pumps on the other hand transport ions actively. The most important ions in the communication process of neurons are potassium ions ($K^+$), sodium ions ($Na^+$) and calcium ions ($Ca^{2+}$). Ion pumps are constantly pumping out $Na^+$ ions and pumping in $K^+$ ions. In exchange for 3 $Na^+$ ions being pumped out, 2 $K^+$ ions are being pumped in [5]. When the neuron is in a resting state $K^+$ ions are found in the interior of the neuron while $Na^+$ and $Ca^{2+}$ ions are found in the exterior of the neuron. By changing the concentration of the ions inside the neuron, the membrane potential of the neuron can be changed. Initially the membrane potential of a neuron has a resting potential of -70 mV [5]. Another important level of the membrane potential is the threshold potential. The threshold potential lies around -55 mV and causes certain ion channels to open. The last important level of membrane potential is the action potential. The action potential is a large positive spike in the membrane potential that can reach +40 mV [5] and that triggers the activation of the neuron.
The eventual activation of the neuron depends on the signals received by other neurons. When the chemical receptors in the dendrite spines of a neuron bind with neurotransmitters received by the axon terminals of other neurons the membrane potential of the neuron either depolarises or repolarises, depending on the type of chemical receptor. Depolarization means that the membrane potential becomes less negative, while repolarisation means that the membrane potential becomes more negative. If the binding of neuro-

transmitters causes the membrane potential to depolarize until the threshold potential is reached, ion channels for $Na^+$ ions in the axon membrane are opened. This allows $Na^+$ ions to enter the axon, causing a further depolarization, which in its turn causes more $Na^+$ ion channels to be opened. When the membrane potential becomes positive the ion channels for the $Na^+$ ions start to close and the ion channels for $K^+$ ions open, allowing $K^+$ ions to leave the neuron. The membrane potential continues to grow positively until all $Na^+$ ion channels are closed. After this peak in the membrane potential, the $K^+$ ions continue leaving the neuron and will cause a hyperpolarisation before the membrane potential returns to the resting potential. Hyperpolarisation happens when the membrane potential becomes more negative than the resting potential. The rising of the membrane potential to a peak and returning to its resting potential afterwards is called the action potential.
When the action potential occurs at the axon hillock it travels further through the axon by opening neighbouring $Na^+$ ion channels. When the action potential reaches the axon terminals it triggers $Ca^{2+}$ ion channels to open, allowing $Ca^{2+}$ ions to flow into the axon terminal. The increased concentration of $Ca^{2+}$ ions causes the containers of neurotransmitters to release the neurotransmitters to the dendrite spine of a connected neuron. When the membrane potential becomes positive the ion channels for the $Na^+$ ions start to close. The membrane potential continues to grow positively until all $Na^+$ ion channels are closed. Afterwards ion pumps actively start transporting the $Na^+$ ions out of the neuron to create a repolarization. Furthermore ion channels for the $K^+$ ions inside the neuron open and allow the $K^+$ ions to leave the neuron. This increases the repolarization and eventually causes a hyperpolarization before the membrane potential returns to the resting potential.

## 2.2 Brain Regions

The brain is structured in different regions that each handle specific brain functions. These brain function are thus associated and mapped to certain areas of the brain. Information about which neural functions are linked to which specific parts of the brain can be embedded in brain atlases. Brain atlases can be thought of as a map of the brain and are often represented by images of different slices of the brain or even by full 3d models. The images or the 3d model that make up a brain atlas are often obtained by MRI scans or slices of a brain that are being stained to visualise the structure of the

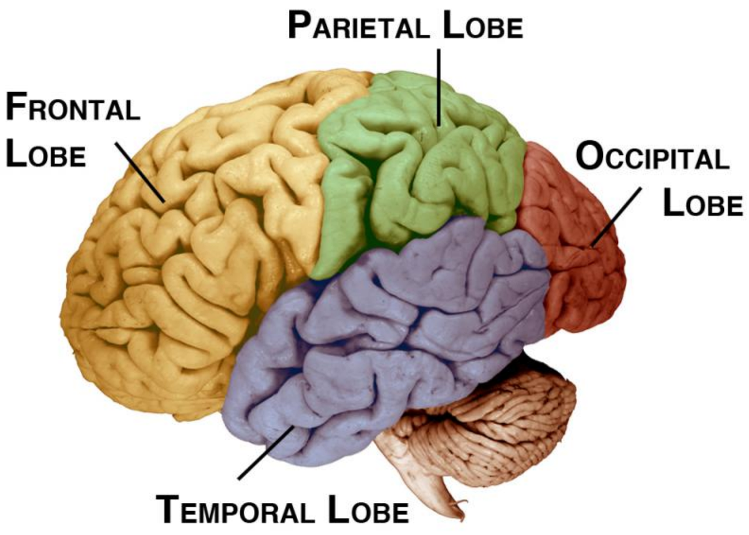


Figure 2.2: The four major lobes of the human brain[1]

slice. The human brain is classified into four lobes, as illustrated in Figure 2.2. These lobes are the frontal lobe, the parietal lobe, the temporal lobe and the occipital lobe. Each lobe can be divided into further areas that are associated to specific brain functions. Such an area is called a functional area. Figure 2.3 depicts the human brain and its functional areas. For each of these functional areas of the brain we will provide a description of the associated functions.

### 2.2.1 Frontal Lobe

The frontal lobe lies — as the name suggests — at the front of each cerebral hemisphere. It is positioned anterior to the parietal lobe. This means that the frontal lobe is positioned in front of the parietal lobe. Furthermore, the frontal lobe lies both superior and anterior to the temporal lobes. This means that the frontal lobe is positioned on top of and in front of the temporal lobes. The frontal lobe can be further divided into areas. These areas are named the prefrontal cortex, the premotor cortex, the primary motor cortex and Broca's area.

[1] Source: `http://serendip.brynmawr.edu/exchange/files/authors/faculty/295/lobes2.jpg`
Last access: 15-06-2014

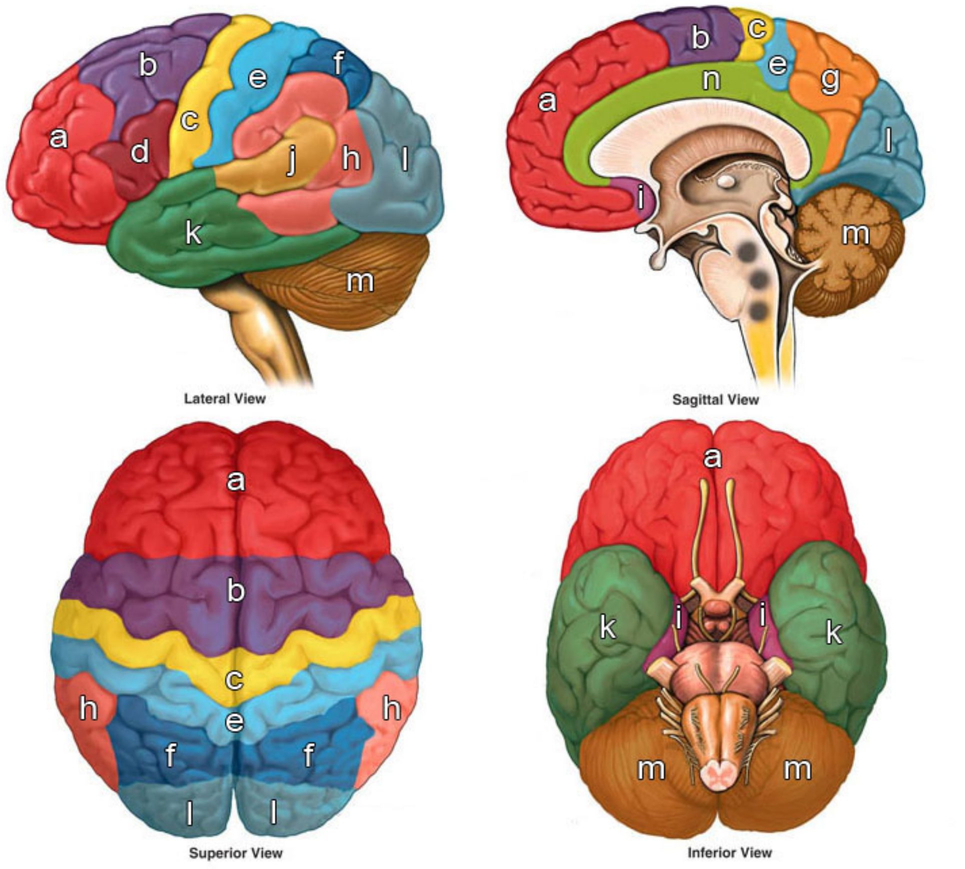


Figure 2.3: Functional areas of the human brain[2]

The prefrontal cortex, as illustrated in Figure 2.3(a), is positioned anterior to the orther areas of the frontal lobe. It is associated to higher mental functions such as creativity, concentration, planning and judgment. Another function that is associated to the prefrontal cortex is emotional expression, which should not be confused with emotional feeling.
The next area of the frontal lobe we discuss is the premotor cortex, depicted by Figure 2.3(b). The premotor cortex lies posterior to the prefrontal cortex. Furthermore it is positioned anterior to the primary motor cortex and both anterior and superior to Broca's area. The premotor cortex is associated to motor functions that involve the orientation and movement of the eyes.

[2] Based on image source: `http://healthfavo.com/wp-content/uploads/2013/09/parts-of-the-brain-lobes.jpg`
Last access: 15-06-2014

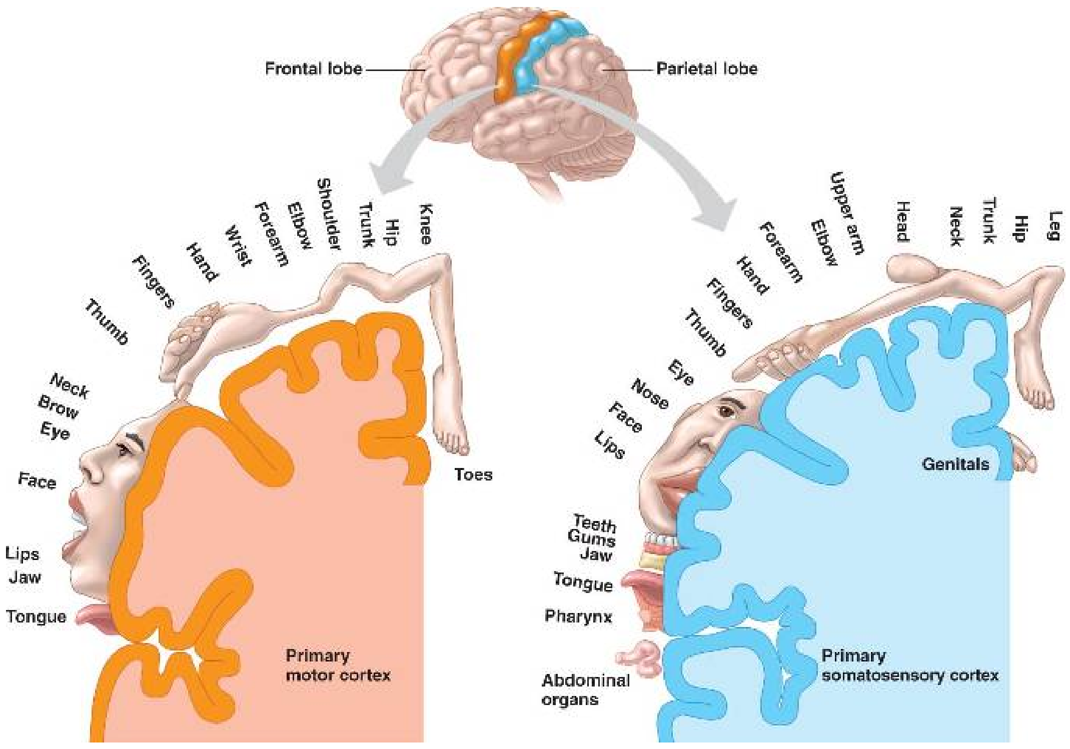


Figure 2.4: The primary motor cortex and the somatosensory cortex along with the mapping of bodyparts[3]

Another area of the frontal lobe is Broca's area. Broca's area is illustrated by Figure 2.3(c) and positioned in between the other areas of the frontal lobe. The brain functions associated to Broca's area involve motor functions for the muscles that are needed to produce sound and speech.

The last functional area of the frontal lobe we discuss is the primary motor cortex, portrayed in Figure 2.3(d). This functional area is positioned posterior to all other functional areas of the frontal lobe. The primary motor cortex is responsible for the movement of most of the muscles and is organised in such a way that its part on the left cerebral hemisphere is responsible for the muscles in the right part of the body, while the part of the primary motor cortex in the right cerebral hemisphere is responsible for the muscles in the left part of the body. Both the left and the right part of the primary motor cortex are organised in such a way that movements of the foot are produced in the top-most part of the primary motor cortex and that movements

[3] Source:`http://bio1152.nicerweb.com/Locked/media/ch49/49_16-MotorSensoryCorts-L.jpg`
Last access: 15-06-2014

of the tongue are produced in the bottom part of the primary motor cortex. All muscles between foot and tongue are organised accordingly between the top and the bottom of the primary motor cortex. Figure 2.4 illustrates the mapping of bodyparts on the cortex.

### 2.2.2 Parietal Lobe

The parietal lobe is positioned posterior to the frontal lobe and superior to the occipital lobe. Thes functional areas that are part of the parietal lobe are the primary somatosensory cortex, the posterior parietal cortex, the precuneus and part of Wernicke's area.
The primary somatosensory cortex lies anterior to the posterior parietal cortex and superior to Wernicke's area, as illustrated in Figure 2.3(e). The primary somatosensory cortex is associated with the sense of touch. Just like the primary motor cortex in the frontal lobe, the part of the primary somatosensory cortex in the left cerebral hemisphere is responsible for the sense of touch in the right part of the body while the part of the primary somatosensory cortex in the right cerebral hemishpere is responsible for the sense of touch in the left part of the body. Analog to the primary motor cortex as well is the organisation of body parts with the sense of touch for the toes located at the top of the primary somatosensory cortex and the tongue located at the bottom. The body parts between the toes and the tongue are organised accordingly from top to bottom. The primary somatosensory cortex and the primary motor cortex are thus similar in the way they organise the body parts. However, notice in Figure 2.4 that the primary somatosensory cortex is responsible for additional body parts in comparison to the primary motor cortex. These additional body parts are body parts that do not contain muscles but are able to send signals to the brain that can be translated to a sense of touch. Such body parts are genitals, teeth gums and abdominal organs. Furthermore, the primary somatosensory cortex lies right next to the primary motor cortex. This is probably not accidental since the primary somatosensory cortex can be seen as a brain area that receives input from the rest of the body while the primary motor cortex gives output to the rest of the body.
Another functional area in the parietal lobe is the posterior parietal cortex, depicted in Figure 2.3(f). This area is positioned posterior to the primary somatosensory cortex and superior to Warnicke's area. The posterior parietal cortex receives input from the cerebellum and the three sensory systems in the brain which are the visual system, the auditory system and the somatosensory system. The input of these three sensory systems and the cerebellum is combined to plan movements. This means that the posterior parietal cortex

is aware of the position of the body parts, which is called proprioception, and the positions of objects external to the body in order to plan movements in the context of the body's environment. Another function of the posterior parietal cortex is recognizing objects based on input provided by the primary somatosensory cortex. This input includes information like weight, texture and temperature. Because the posterior parietal cortex attempts to recognize objects based on input of the primary somatosensory cortex it is sometimes called the somatosensory association cortex.
Next we discuss the precuneus. This part of the parietal lobe lies medial in both hemispheres and is illustrated by Figure 2.3(g). This means that the precuneus is positioned on the cortex area in between both hemisperes. The precuneus lies posterior to the primary somatosensory cortex, anterior to the occipital lobe and inferior to the posterior parietal cortex. The functions related to the precuneus are those of self-awareness and self-consciousness, consciousness in general, and visuo-spatial processing. With self-awareness and self-consciousness we mean the judgement of one's own personality and actions in regard to other people or compared to the personality and actions of other people. The next function, consciousness in general, means the consciously experiencing or gathering information from our environment. Visuo-spatial processing is the last responsibility and is related to a function of the posterior parietal cortex. It involves shifting attention from one spatial location to another when planning or performing movements of the body.
The last functional area in the parietal lobe is part of Wernicke's area. We say part of because Wernicke's area is divided across both the parietal lobe and the temporal lobe. Figure 2.3(h) shows Wernicke's area, which is associated to the comprehension of written and spoken language.

### 2.2.3 Temporal Lobe

The temporal lobe lies inferior to, which means below, the frontal lobe and the parietal lobe, and anterior to the occipital lobe. The temporal lobe consists of the olfactory cortex, the primary auditory cortex, the medial temporal cortex, an association area and a part of Wernicke's area.
Figure 2.3(i) shows the olfactory cortex. The olfactory cortex lies right above the nasal cavity and inferior to the frontal lobe. The olfactory cortex is responsible for processing odors. The primary auditory cortex lies right beneath the parietal lobe and is responsible for the identification of sound.
The next functional area of the temporal lobe we discuss is the primary auditory cortex, as illustrated by Figure 2.3(j). The primary auditory cortex contains neurons that are each responsible for a specific frequency. Furthermore, these neurons are organised in the primary auditory cortex from low

to high frequencies as well. Another function of the primary auditory cortex is to locate sounds in space.
Next, we discuss the medial temporal cortex. The medial temporal cortex is sometimes referred to as the medial temporal lobe. The medial temporal cortex or lobe is located at the inside of the temporal lobe in each cerebral hemisphere. The medial temporal cortex is involved in the formation, storage and access of declarative memory. Declarative memory is long-term memory that can be consciously recalled. Therefore, declarative memory can be further divided into two types of memory. These two types of declarative memory are semantic memory and episodic memory. Semantic memory is memory that contains general knowledge and facts we have come to known about the world, while episodic memory holds memories of moments and context of an event that occured in a specific place where we have been at a specific point in time.
Another functional area of the temporal lobe we will call the association area for ease of use. The association area, as depicted by Figure 2.3(k), actually consists of three gyri which are the superior temporal gyrus, the middle temporal gyrus and the inferior temporal gyrus. A gyrus is basically a ripple in the cerebral cortex. The function of this group of gyri is the recognition of objects and faces based on visual information and the perception of expressions and emotions based on facial stimuli. For the recognition of objects and faces, the association area of the temporal lobe receives visual information from the occipital lobe, such as shape and color, to identify an object or a face based on the recollection of memory about known objects and faces. This identification process is in a way similar to the process in the somatosensory association cortex in the parietal lobe where objects are identified from memory based upon information about texture, weight and temperature, provided by the primary somatosensory cortex.
The temporal lobe also contains part of Wernicke's area, which is illustrated by Figure 2.3(h). The function of Wernicke's area is the comprehension of written and spoken language, as already mentioned in the subsection on the parietal lobe 2.2.2.

### 2.2.4 Occipital Lobe

The occipital lobe, as depicted by Figure 2.3(l), is located inferior to the parietal lobe and posterior to the temporal lobe. The functional regions that the occipital lobe consists of are called visual areas and are denoted by the codes V1, V2, V3, V4 and V5. The visual area V1 is also called the primary visual cortex. The visual areas in the occipital lobe are connected with each other and share information in a feedforward and feedback system. In connection, the visual areas can process information about the movement of objects. Information about these movements includes the direction, the speed and the motion of the movement of the object. Furthermore, the visual areas can process information about the spatial frequency and temporal frequency — which is basically speed — of objects. By combining information about the spatial and temporal frequency, the visual areas of the occipital lobe are able to derive spatiotemporal characteristics of objects. This spatial, temporal and spatiotemporal information enables pattern recognition. Another function of the visual areas of the occipital lobe is keeping track of visual characteristics of objects — such as shape and color — and minor changes that occur within them.

### 2.2.5 Cerebellum

The cerebellum, as illustrated by Figure 2.3(m), is not part of the cerebrum and therefore is not part of any of the four major lobes. However, the cerebellum is part of the human brain and is responsible for functions as well. The major responsibility of the cerebellum is the fine-tuning of coordination, precision and accurate timing of movements. The cerebellum is also involved in keeping the body in balance and the posture of the body. Finally, the cerebellum also plays a role in motor skill learning. Although the actual learning process may not occur in the cerebellum itself, the cerebellum is thought to provide certain signals necessary for the learning process of precision movements.

### 2.2.6 Cingulate Cortex

The cingulate cortex is an area in the brain that is not part of one of the four major lobes of the human brain. However, some of the functions of the cingulate cortex are still worth mentioning in this section. Figure 2.3(n) shows the cingulate cortex. The cingulate cortex lies on the medial surface of each hemisphere, inferior to both the frontal and the parietal lobe. The cingulate cortex is further divided in the anterior cingulate cortex and the posterior cingulate cortex.

The anterior cingulate cortex is involved in learning based on error detection, self-consciousness of emotion and registering pain. The learning function of the anterior cingulate cortex evaluates thoughts and actions and alters memory to avoid errors in the future. As mentioned, the anterior cingulate cortex is also involved in self-consciousness of emotion. This means that the anterior cingulate enables us to be aware of our own emotions. The last function of the anterior cingulate cortex is the registering of pain. However, the anterior

cingulate cortex is more involved in the emotional reaction to pain and the intensity of pain rather than the perception of pain itself.
The second part of the cingulate cortex is the posterior cingulate cortex. The function of the posterior cingulate cortex is emotional response. This emotional response involves generating an emotional reaction to certain stimuli or to the recollection of auto-biographical memories.

## 2.3 Brain Rhythms

When brain activity is measured this produces a wavelike function which has different frequencies and different amplitudes for these frequencies. There is a certain correlation between these frequencies and specific brain activity. It appears that for certain types of brain activity, or even certain types of state of mind, there are specific frequency ranges in which the amplitudes of the frequencies in this range are higher than others. These frequency ranges are also called rhythms where each rhythm is assigned a Greek letter. In this section we will provide an overview of all brain rhythms and their associated brain activity in order of increasing frequency. The first rhythm is the delta rhythm. This rhythm has a frequency range containing all frequencies lower than 4 Hz. The delta rhythm thus contains slow waves. However, these waves usually have a high amplitude. This rhythm typically occurs when the subject is in a deep sleep [35].
Second, there is the theta rhythm. The theta rhythm has a frequency range of 4 Hz to 7.5 Hz. This rhythm is associated to sleeping [35].
Next, we have the alpha rhythm which emerges itself in frequencies between 7.5 Hz and 13 Hz. The alpha rhythm is associated to relaxation and closing the eyes [17][35]. This means that the alpha rhythm occurs when few other mental tasks are being performed. Furthermore, the alpha rhythm disappears from the moment a subject starts to concentrate or starts to actively think. Stress can disrupt the alpha rhythm as well. Because of this disruption, the alpha rhythm can be used to detect when a subject is between performing mental tasks and to detect the moment when the subject starts performing a new mental task.
Following the alpha rhythm we have the beta rhythm. The beta rhythm has a frequency range of 13 Hz to 30 Hz. The beta rhythm is typically lower in amplitude compared to rhythms like the alpha rhythm and delta rhythm. Furthermore, the beta rhythm has especially low amplitudes in varying frequencies for brain activity which involves active thinking, concentration and anxiety [35]. Low beta activity is also associated with the movement of muscles [35].

Due to the fact that each of these brain rhythms are associated to specific brain activity and a specific frequency range, they can be of big aid in the detection of brain activity if the brain rhythms related to the brain activity are known beforehand. We will discuss this further in Chapter 3 on Brain Computer Interfaces.

# 3 Brain-Computer Interfaces

Brain-Computer Interfaces (BCI) are interfaces that form the link between an application and the brain activity of the person that uses the application. The idea is to measure signals from the user's brain and analyse these brain signals to detect what mental tasks the user is performing. When a specific mental task is detected it can be used as input for the application. For example, imagining movements of the left and right arm can be used to move an object left or right in a virtual reality application.

This chapter discusses the difference between an online and an offline Brain-Computer Interface, in Section 3.1, and describes the different components to accomplish a BCI. The task of the first component is to measure the brain activity itself. Different techniques for measuring brain activity are described in Section 3.2. EEG is one of the techniques to measure brain activity and we will focus on EEG-based BCI for the remainder of the chapter. EEG brain signals typically have a bad signal-to-noise ratio. Therefore, the second step in an EEG-based BCI is to improve the signal-to-noise ratio of the EEG signal by using preprocessing techniques. The preprocessing component is described in Section 3.3. When the signal-to-noise ratio is optimized, the EEG signal is passed on to the feature extraction component. The feature extraction component extracts meaningful data out of the EEG signal in order to reduce the data dimensionality. The idea is to maintain only the information in the EEG signal that is relevant for the classification of the mental task that is related to the measured EEG signal. The feature extraction step is described in Section 3.4. The last component of a BCI is the classification component which recognises the mental task related to a chunk of an EEG signal based on the extracted features. The classification step is described in Section 3.5

## 3.1 Online Versus Offline BCI

Brain-Computer Interfaces can either operate online or offline.
An offline BCI measures the brain signals beforehand and returns the classification results later in time. This allows techniques in each step of the BCI to analyse the brain signal as a whole, meaning that, for a given time point in the recording of the brain signal, the techniques can take brain signals at a later point in the recording into account as well. Furthermore, noise in the brain signals can be detected and marked by hand. The performance in computing time is also less crucial than for online BCI.
An online BCI on the other hand will classify incoming brain signals in real-time. This means that the techniques used in the preprocessing step, the feature extraction step and the classification step need to do their calculations on the fly, which means that the different steps of the BCI can not rely on brain signals measured after the current brain signal. Also, the brain signals can not be controlled by hand, so all steps in the BCI need to happen automatically. Furthermore the calculations for each technique in each step of the BCI can not take up too much time since the classification result should be presented as quick as possible. This means that the used techniques in each step of the BCI should have a decent performance in computing time. For some techniques it is not even possible to use them in online BCIs.
Offline BCIs are mostly used to analyse patients since the results of an analysis are not required to be produced in real-time. Online BCIs are mostly used in applications where the classification results of the BCI to control an object in a virtual reality or a machine in the real world.

## 3.2 Data Acquisition

### 3.2.1 Electroencephalography

Electroencephalography (EEG) uses electrodes that are placed on the scalp to measure the electric field generated by the ionic flows within neurons. As described in Section 2.1, these ionic flows occur when neurons communicate with each other and are largest when an action potential occurs. When ions are pumped out of a neuron they push away similarly charged ions, which

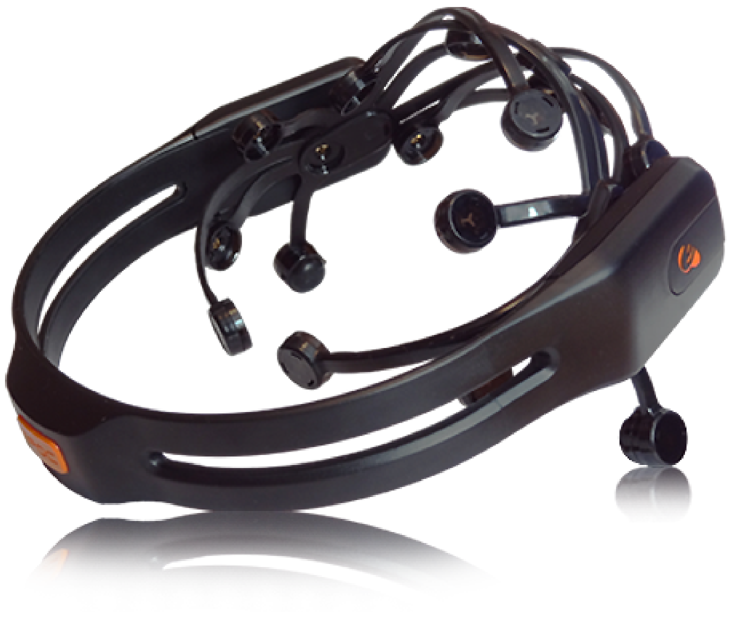


Figure 3.1: The Emotiv Epoc EEG capture device[1]

in turn pushes away other similarly charged ions. This can be seen as waves of ions pushing each other forward. The electrical field generated by such a wave of ions can be measured when nearing the scalp by electrodes. These electrodes are made of metal and contain electrons that are pushed or pulled by the electrical field of the ions. The voltage generated by the electrons being pushed or pulled can be measured by a voltmeter in regard to another electrode. Therefore, EEG requires at least one reference electrode that is not directly used in the measurement but for the comparison to other electrodes.

Because the electric field generated by ions being pumped out of a single neuron is far too small to pick up, thousands to millions of neurons need to fire synchronously to be able to measure their synchronised activity. Furthermore, the ions being pumped out of the neurons need to line up to be able to create waves, which means that the synchronously firing neurons need to have the same spatial orientation. Based on the characteristics posed to neurons to have synchronous activity and similar spatial orientation, it is believed that the main contributors to EEG signals are pyramidal neurons. Pyramidal neurons gained their name from the triangular shape of their soma or cell body. Pyramidal neurons are mainly found in the cerebral cortex.

[1] Source: `http://juliakester.com/images/processImages/emotivlarge.png`
Last access: 15-06-2014

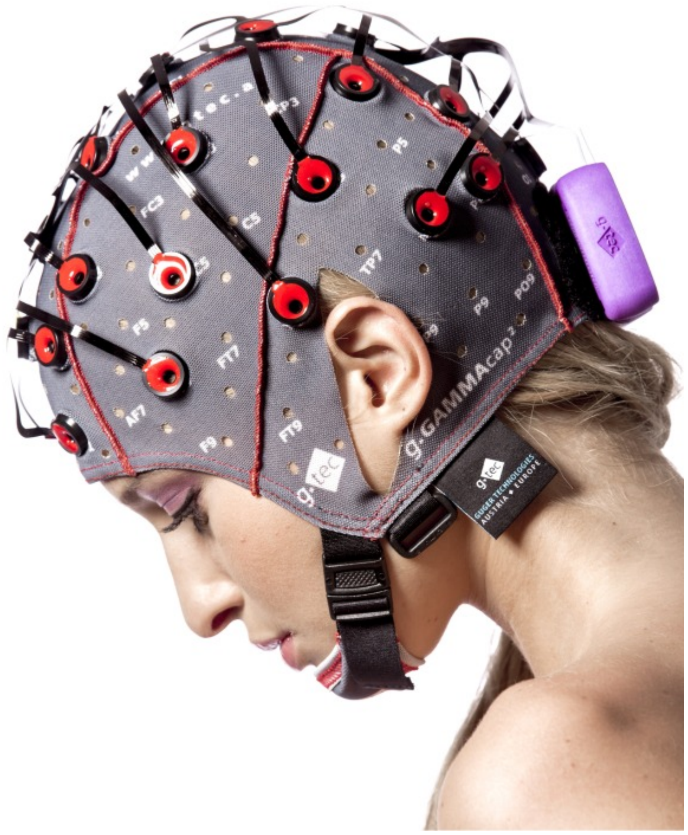


Figure 3.2: EEG capture device using a cap for electrode placement[2]

Lastly, EEG signals can most easily be detected from neurons that lie near the scalp. Suppose we have a point $p$ in an electric field at a distance $r$ from the charge $q$, which is the source of the electric field. According to Coulomb's Law the electric power $E$ in point $p$ induced by the charge $q$ is given by equation 3.1 with $k_e$ being Coulomb's constant.

$$E = k_e * q/r^2 \tag{3.1}$$

Coulomb's Law tells us that the power in a point of the electric field weakens with the square of distance. The power at the electrodes on the scalp of electric fields generated by ions deeper in the brain is therefore too weak to be measured.

[2] Source: `http://www.gtec.at/var/plain_site/storage/images/media/images/products/g.nautilus/nautilus_sideview/102865-1-eng-GB/nautilus_sideview.jpg`
Last access: 15-06-2014

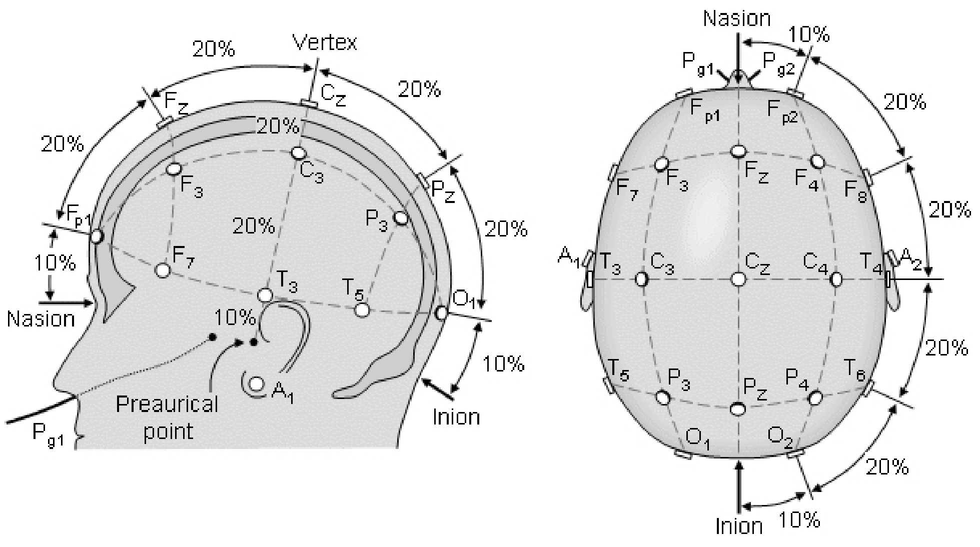


Figure 3.3: Placement, naming and spacing of electrodes following the 10-20 system [29]

A typical EEG capture setup consists of a set of electrodes, an amplifier, a voltmeter and an acquisition device. The acquisition device is a computer system that will acquire the signals and send them through to the BCI system. The BCI and the acquisition device can be the same computer system. Some EEG capture devices integrate the electrodes, amplifier and voltmeter into one device, like the Emotiv Epoc for instance. The Emotiv Epoc device is depicted in Figure 3.1. Other EEG capture setups might work with a cap that goes on the head and allows for electrodes to be pinned upon it, as shown in Figure 3.2. Electrodes of an EEG capture device are placed on precise locations and each electrode is given a name for the specific location they are placed on. One way of placing and naming the electrodes accordingly is the 10-20 system [16]. The 10-20 system, or international 10-20 localisation, is an international standard for the placement of EEG electrodes. The placement of electrodes and their names dictated by the 10-20 system can be seen in Figure 3.3. The numbers 10 and 20 in the name of the 10-20 system refer to the spacing between the electrodes. This spacing between the electrodes is 20% of the total distance measured from the nasion to the inion and the total distance measured from the left preauricular point to the right preauricular point. Furthermore, the outer electrodes are spaced 10% from the nasion, inion, or preauricular points. To provide for the need for placing more electrodes on more locations the 10-20 system was extended tot the 10-10 system[30]. The placement and naming of electrodes according to

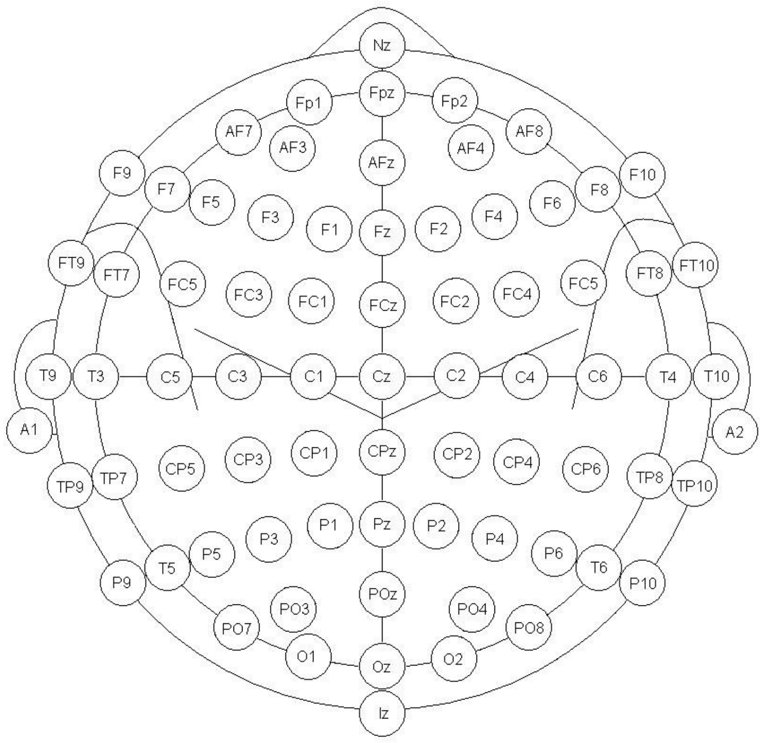


Figure 3.4: Placement and naming of electrodes following the 10-10 system[3]

the 10-10 system can be seen in Figure 3.4.

### 3.2.2 Magnetoencephalography

Magnetoencephalography (MEG) measures brain activity based on the ionic flows that are created by neurons. These ionic flows form the basis for EEG measurements as well. The difference between EEG and MEG in regard to the source signal is that EEG measures the electric field generated by the ionic flows, while MEG measures the magnetic field that results from the electric field generated by the ionic flows. According to Maxwell's equations, and in particular the fourth equation which is Ampere's circuital law with an enhancement of Maxwell, an electric current will produce a magnetic field that is oriented orthogonally to the orientation of the electric current.
Apart from both being based on the ionic flows of neurons, another similarity of MEG and EEG are the requirements for neurons in order to pick up their

[3] Source: `https://wiki.umms.med.umich.edu/download/attachments/90734989/EEG+10-20+system+map.JPG`
Last access: 15-06-2014

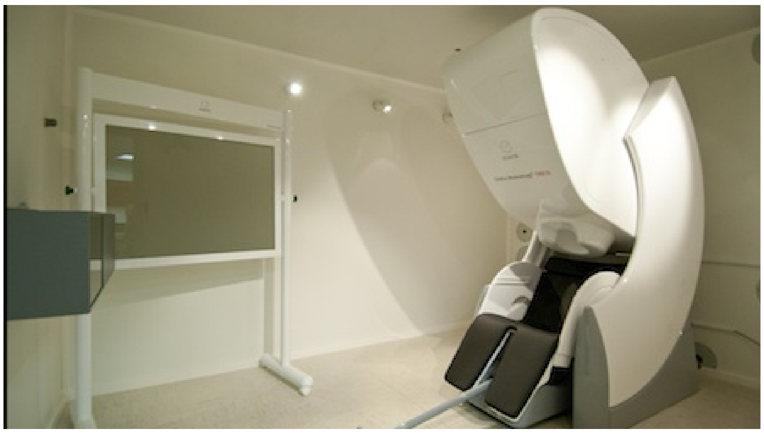


Figure 3.5: A magnetic shielded room with an MEG measuring machine[4]

activity. Neurons need to have synchronised activity in groups of thousands to millions and they need to be spatially oriented in the same way. This means that the neurons that are the source of the MEG signal are believed to be pyramidal neurons as well, just like the EEG signal.

Because the power of magnetic fields in a normal environment is far greater than the magnetic fields induced by the brain, MEG needs to be recorded in a magnetic shielded room. Figure 3.5 contains a picture of such a magnetic shielded room with an MEG measuring machine. The machine itself has a helmet-shaped part to fit the head of the subject. This helmet-shaped part contains hundreds of SQUID sensors that are arranged into arrays. A SQUID sensor is a very sensitive magnetometer that can measure weak magnetic fields.

It was believed that the resistivity of the skull and scalp affected EEG more than MEG, which would lead to EEG being spatially less accurate than MEG. However, Malmivuo and Suihko have shown that EEG has in fact a better spatial resolution than MEG [23]. Furthermore, the equipment needed to measure MEG is far more expensive than the equipment to measure EEG. EEG is also quite portable, while MEG uses a rather big and heavy machine and needs a magnetically shielded room. Liu et al have shown that the most accurate results can be achieved by combining EEG and MEG signals, providing that the optimal scale factor for both is known [20].

[4] Source: `https://www.floridahospital.com/sites/default/files/whatismeg1.jpg`
Last access: 15-06-2014

Figure 3.6: Image of an MRI scanner[5]

### 3.2.3 Functional Magnetic Resonance Imaging

Functional Magnetic Resonance Imaging (fMRI) should not be confused with traditional MRI. An MRI scan creates an image of the body based on different tissue in the body, whereas an fMRI scan creates several images that represent activity in the brain over time. The process of creating an MRI scan and an fMRI scan differs as well. Although both scans use an MRI scanner (Figure 3.6) that generates two magnetic fields of different strengths around the human body, the process of creating an MRI scan and an fMRI scan differs as well.

An MRI scan uses the fact that the human body is full of water. A water molecule consists of two hydrogen atoms and an oxygen atom. The hydrogen atom can be considered as the simplest of all atoms with only a single proton in its nucleus. A proton has a positive pole and a negative pole and spins around the axis between these poles, just like the earth. The spinning movement of the proton in each hydrogen atom nucleus can be influenced by the generated magnetic fields of the MRI scanner. The altered spinning movement of each proton happens at a certain frequency depending on the environment the water is in. An MRI scan observes the magnetic field generated by the altered spinning movement of each proton and the frequency of the spinning movement to determine differences in the environment the water is in. Therefore, MRI can detect the differences in tissues within the

[5] Source: `http://mms.businesswire.com/bwapps/mediaserver/ViewMedia?mgid=294480&vid=4&download=1`
Last access: 15-06-2014

body.
Because an MRI only observes differences in structure a different process is used in fMRI to observe changes in brain activity. When neuronal activity increases in a specific part of the brain, the involved neurons need glucose to provide them with energy. Because the brain does not store glucose the blood needs to transport glucose to the brain. Glucose is transferred into energy by a burning process that requires oxygen. This means that the blood needs to transport oxygen as well to the part of the brain where neuronal activity increases. The transport of glucose and oxygen by the blood to active neuronal areas is known as the Haemodynamic response. Blood uses hemoglobin to transport oxygen by binding oxygen to these hemoglobin molecules. When hemoglobin does not have bound oxygen it is more attracted to magnetic fields. In an fMRI scan changes in the attraction of the hemoglobin molecules to the magnetic fields of the MRI scanner are observed in order to determine where oxygen-rich blood is flowing through the brain. Signals resulting from an fMRI scan are called blood oxygenation level-dependent (BOLD) signals.

### 3.2.4 Functional Near-Infrared Spectroscopy

Functional Near-Infrared Spectroscopy (fNIRS) is, just like fMRI, based on the Haemodynamic response. Where fMRI uses the magnetic attractiveness of deoxygenated-hemoglobin to observe brain activity, fNIRS uses the specific absorption spectra of oxygenated- and deoxygenated-hemoglobin in the near-infrared light range. The near-infrared light, or NIR, range lies between 700 nm to 1000 nm. The values in nanometer denote the wavelength of the light. The wavelength of light changes when travelling through different matter due to absorption by the matter. For different wavelengths of light a type of matter has different absorption coefficients. The absorption coefficients of skin, tissue and bone are very low for wavelengths of NIR, while the obsorption coefficients of oxygenated- and deoxygenated-hemoglobin are much higher. This means that skin, tissue and bone are rather transparent to NIR light and that oxygenated- and deoxygenated-hemoglobin can be easily detected because of their attenuation of the NIR light. The difference between oxygenated- and deoxygenated-hemoglobin can be made due to their different absorption coefficients for different wavelenghts of NIR light. By monitoring the changes in the presence of oxygenated- and deoxygenated-hemoglobin over time, a functional image can be created.
The device to measure fNIRS signals itself uses emitters of NIR light that send NIR light into the brain. Detectors that are placed near an emitter pickup the NIR light that travelled through a portion of the brain. Figure 3.7 shows an fNIR measurement device. Notice that the difference between

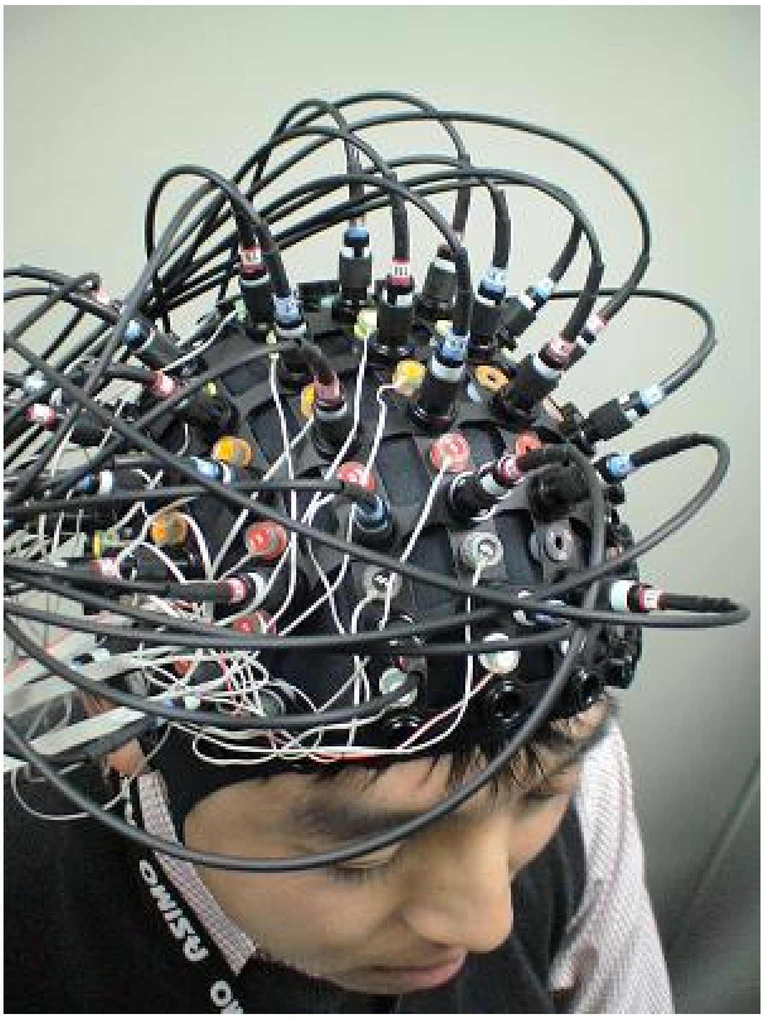


Figure 3.7: An fNIRS measuring device[6]

emitters and detectors is represented by the colors red and blue. Figure 3.8 shows how the light travels from an emitter through the brain to a detector. The light sent by the emitters does not travel deep enough to the brain and the detectors also can not detect the light too deep in the brain. Therefore, fNIRS can not be used to measure brain activity from deeper sources in the brain. Furthermore, signals obtained with fNIRS typically have a delay of several seconds [39].

[6] Source: `http://www.cns.atr.jp/~dcallan/img/nirs.jpg`
Last access: 15-06-2014

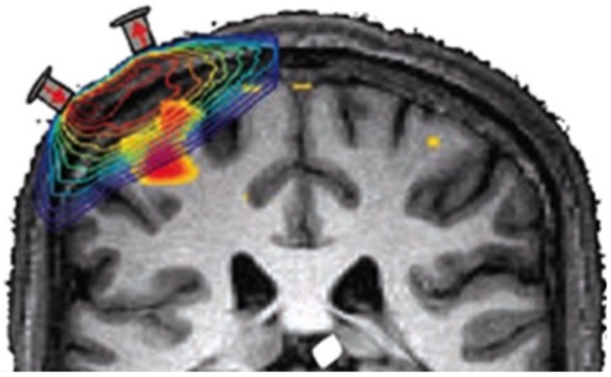


Figure 3.8: The near-infrared light sent by the emiters of an fNIRS measuring device is bent towards neighbouring detectors[7]

## 3.3 Preprocessing

EEG data is typically contaminated by so called artefacts. These artefacts can be viewed as noise and thus are the reason EEG data has such a poor signal-to-noise ratio. In order to improve this poor signal-to-noise ratio some signal processing techniques exist to remove these artefacts from the EEG data. In this section we discuss the types of artefacts in EEG data and some preprocessing techniques that can imporve the signal-to-noise ratio of the EEG signal.

### 3.3.1 Physiologic Artefacts

A physiologic or biologic artefact has a source —other than the brain— that is located within the body of the BCI user. Physiologic artefacts include eye movement, muscle activity and heart beats, tongue movement, blood pulse and skin potentials.
The first type of artefacts we discuss are called *electro-oculogram (EOG) artefacts*. EOG artefacts result from eye movements and eye blinks. Eye movements are picked up by the electrodes of an EEG capture device because the eyeball acts as a dipole, with a positive pole oriented at the cornea and a negative pole oriented at the retina. When the eyeball moves, the movement of the poles of the eyeball generates an alternate current field with a large amplitude. Lateral eye movement affects electrodes F7 and F8 (Figure 3.4) the most, while electrodes Fp1 and Fp2 (Figure 3.4) are the most

[7] Source: `http://www.researchimaging.pitt.edu/sites/default/files/location_images/9b.jpg`
Last access: 15-06-2014

affected by vertical eye movement. Eye blinks are considered as vertical eye movements as well since an eye blink causes the eye to look upwards. This is know as the Bell phenomenon. Both eye blinks and eye movement artefacts are dominant in frequencies below 4Hz [10][3].
artefacts originating from muscle activity are called *electromyogram (EMG) artefacts*. EMG artefacts originate mostly from the muscle activity in the faces, which is typically picked up mostly by the Fp1 and Fp2 electrodes[15]. EMG artefacts are dominant in frequencies between 20Hz and 60Hz [10][3]. Because the potential of EMG artefacts is of shorter duration than potentials generated by neurons in the brain, EMG artefacts can easily be detected based on the shorter duration of their potential. Additional indicators of EMG artefacts can be the shape and frequency of their potentials.
Another type of artefact are *glossokinetic artefacts*, which originate from movements of the tongue. Just like the eyeballs, the tongue acts as a dipole with the tip being a negative pole and the base a positive one. The movement of the tongue moves its poles as well, which generates an alternate current. Glossokinetic artefacts are most presented in channels of artefacts above the frontal lobes and the more frontal areas of the temporal lobes, and are present in the delta frequency band[15].
The next type of artefacts we discuss are *electrocardiogram (ECG) artefacts*. Electrocardiogram artefacts originate from heart potentials. These heart potentials travel across the surface of the scalp and are best observed in electrodes A1 and A2[15]. ECG signals can be recognized by their rhythmicity.
artefacts can also be generated by a blood vessel. When an electrode is placed over a blood vessel a *pulse artefact* might occur. This pulsation might appear as EEG activity because of the slow waves it produces. There is however a direct relation with ECG artefacts. The potential signal of the heart happens approximately 200 to 300 milliseconds before the pulse. This way, the pulse artefact can be detected and filtered out.
*Skin potentials* caused by sweating can also influence the EEG readings. When a user starts sweating the skin releases sodium chloride and lactic acid that reacts with the metals of the electrodes. This produces huge slow waves that might appear as EEG activity[15].

### 3.3.2 Extraphysiologic Artefacts

An extraphysiologic or technological artefact has a source that is outside the body. Such an artefact can be line noise, originating from the power source of the EEG measurement device, noise from electrodes, or environment noise.
The electrodes themselves can also cause artefacts. When an electrode pops it generally creates a sharp peak in the waveform. This is due to an abrupt

change in the impedance. Electrode pops usually occur when electrodes are moved a bit on the scalp. Electrode popping can be observed by a peak in the waveform of one single electrode.
The alternating current of the power source of the EEG measuring device is an artefact as well which is typically a result of pour grounding. Depending on the power source's frequency the artefact is present at either 50 Hz or 60 Hz [15].
Besides artefacts that are related to the EEG capturing device, environment artefacts can occur as well. Environment artefacts are caused by sources around the user and the EEG measuring device. When people other than the user move around the user they create an artefact of both electrostatic and capacitive origin. Another environment artefact is caused by radiation[15]. Cell phones, WiFi, radio, television and many other sources emit waves that cause an overload of the electrodes' amplifier. This results in a contamination of the EEG data.

### 3.3.3 Preprocessing Techniques

We have discussed a variety of artefacts and their characteristics. To filter out the artefacts in the EEG signal in order to improve the signal-to-noise ratio, some techniques can be used. These techniques can also be used to focus on specific parts of the EEG signal in the BCI.

*Spectral filters* can be used to exclude certain frequencies that are characteristic for some artefacts or to focus on a frequency band that is likely to contain the brain activity of interest. In order to filter frequencies a Fast Fourier Transform, FFT, has to be applied to the signal. The Fast Fourier Transform performs a Fourier Transform on the discrete function of a signal to obtain a function that consists of the summation of the different frequencies of the signal, weighted by their amplitude. To exclude certain frequencies their weight can be set to zero after which an Inverse Fourier Transform can be applied to return to the wave function of the signal. Different types of frequency filters exist.
The simplest frequency filter is a notch filter which filters out a single frequency. In EEG signal preprocessing, notch filters are typically used to filter out line noise originating from the power source. The reason for this is that line noise is present at a frequency of either 50 Hz or 60 Hz. Furthermore, it is not unusual for EEG signal measuring devices to already apply a notch filter on the incoming EEG signals.
Another filter is the low pass filter which allows all frequencies below a certain value to pass. Thus it is used to cut off higher frequencies. In EEG

signal processing it is mostly used to remove environment artefacts since the sources of these artefacts usually produce waves that have a high frequency. The exact cut-off frequency depends on the brain activity that is focussed on, but 30 Hz is a value that is often used.
The opposite of a low pass filter is a high pass filter. A high pass filter allows all frequencies above a certain value to pass. It is used to cut off lower frequencies. A high pass filter can successfully remove the slow waves cause by pulse and skin potentials. EMG artefacts are present at lower frequencies as well and can therefore also be removed with a high-pass filter. Caution is required though because low frequencies often contain important data as well. A typical cut-off value for a high pass filter is 0.1 Hz to 0.3 Hz.
Finally, the combination of a low pass and high pass filter is a band pass filter. It allows all frequencies between two values to pass. In the first chapter we mentioned brain rhythms. Brain rhythms are ranges of frequencies that are associated to specific brain activity. For example, sensorimotor activity is found in the beta rhythm, which has a frequency range of 12.5 Hz to 28 Hz. Suppose that we know in what brain frequency range we can find the brain activity we are looking for. We can then use a band pass filter to remove all frequencies that are outside of that range. This way we can remove artefacts as well as other noise or signals of unwanted brain activity.

Another technique next to spectral filtering is *Independent Component Analysis (ICA)*. ICA is a blind source separation technique used to unmix signals that have independent sources. For instance, suppose we have a room where different conversations take place and where microphones are placed on different places. These microphones would each record a different combination of all conversations. However, we would like to have a recording of each conversation separately. This is known as the cocktail party problem. Based on the recordings of the microphones, ICA is able to solve this problem.
However, a correct result of applying the ICA technique is based on some assumptions.

1. The sources of the signal have to be statistically independent.

2. The distribution of values in each source signal are non-Gaussian.

3. For N signal sources, at least N observations have to be made. In the example of the cocktail party problem this means signals have to be recorded by at least N microphones.

Makeig et al [22] showed the usefulness of ICA in the preprocessing of EEG data based on its results and assumptions. ICA takes the EEG data of each electrode as observations. Next it unmixes all EEG data into independent components. By examining these components, their type of signal can be found. If the signal appears to be noise or not of interest, the component can be removed. After all unwanted components are removed, the remaining components can be mixed back together. This is the reason why ICA is such a powerful tool in EEG preprocessing. It can separate artefacts like line noise, eye movement, EMG, ECG and pulse [14] as well as dominant signals in different brain rhythms [22]. A downside of the ICA technique is that an expert has to examine the independent components, meaning that ICA can not be used in an online BCI. However, Nolan et al [28] have developed FASTER, which is an automated algorithm for artefact removal based on ICA. FASTER makes an estimation of different paramaters of various aspects of the EEG data and the independent components. Based on these parameters it decides whether a components contains usefull data or an artefact. Nolan et al [28] proved that the algorithm is rather accurate.

## 3.4 Feature Extraction

EEG data contains a lot of information like spatial, temporal and spectral information. Additionally, typical EEG-based BCI training data does not contain a lot of trials. An overload of information for a single trial and a low amount of trials make it difficult for classification algorithms to determine a correct mapping onto a class. In order to have an accurate classification, we need to reduce the amount of data. This can be done by extracting meaningful features. Features are values that describe certain characteristic aspects of a chunk of data. This way, features hold representative information about the data in a way that not all data has to be maintained.
This section outlines the different techniques that can be used to extract features out of EEG data.

### 3.4.1 Fourier Transform

Spectral features are used a lot in BCI systems. Because of the different brain rhythms and their associated mental states and frequency bands, features extracted from the frequency spectrum typically contain a lot of information about the mental task associated to an epoch [1]. An epoch is a specific part of EEG data that is indicated by a start point in time and an end point in time.
As we mentioned before in the preprocessing section, the frequency spectrum of a wave signal can be found by applying a Fast Fourier Transform. We can

then divide the spectrum into bands of 1 Hz for example and take the amplitude of each band as a feature.
The downside of this technique is that it does not contain temporal information. Features extracted from a Fourier transform contain information about the frequencies of an EEG signal and their strength, but they do not describe the time when these frequencies occur. However, this downside can be overcome by dividing the EEG signal into smaller chunks of the EEG signal. Each of these chunks starts at a specific point in time and ends at a specific point in time. By performing a Fourier Transform on each chunk of EEG data the frequency spectrum of different moments in time can be calculated. Amplitudes for specific frequencies can than be extracted.

### 3.4.2 Band Power

The Logarithmic Band Power technique can be used to extract features containing information about the power of the signal in a specific frequency band or range. The power of a signal is the square of the amplitude in each sample of the EEG signal.
In Section 3.3 we discussed the use of the band pass filter to pass only a specific band of frequencies to exclude artefacts that operate on frequencies outside the specified frequency band. Besides for removing artefacts, the band pass filter can also be used to gain an insight in signal characteristics for certain frequency bands, which is exactly what the Logarithmic Band Power technique will use the band pass filter for. By using several band pass filters the EEG signal can be split according to the specified frequency bands. For each frequency band the power of each sample of the signal can be calculated. Finally the average of the powers of each sample in the frequency band can be taken to represent the power of the whole frequency band. These power averages can be used as features to make a distinction between more or less powerful frequency bands.

### 3.4.3 Wavelet Transform

As opposed to a Fourier Transform, a wavelet transform does not only handle spectral information but temporal information as well, as motivated by [31][38]. A wavelet is an oscillation with a fixed frequency and an amplitude that starts at zero, increases towards a maximum in the middle and then decreases again to zero as illustrated in Figure 3.9. When performing a wavelet transformation, convolution is applied to a wavelet and another signal, in this case the EEG signal. This produces a third function, the convolution, that represents the overlap of the area between the wavelet and the EEG

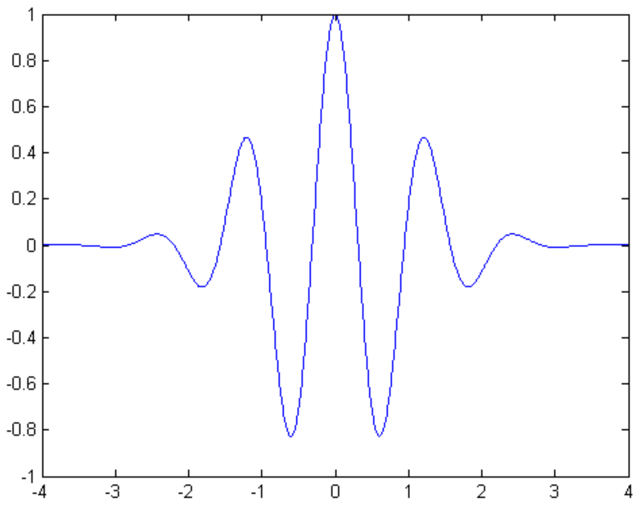


Figure 3.9: A wavelet[8]

signal, where the wavelet is slid over the EEG signal. The convolution of a signal $f$ over a signal $g$ is given by equation 3.2.

$$(f * g)(n) = \sum_{i=-\infty}^{\infty} u(i)(n-i) \tag{3.2}$$

A wavelet transform uses a scale factor and an offset. The scale factor can alter the frequency of the wavelet and thus either dilate or compress the wavelet. The offset determines the offset of the wavelet in relation to the signal. This way the wavelet can match each frequency of the EEG signal on each time value. This is the reason why wavelet transforms contain spectral information as well as temporal information. This information is represented by so called wavelet coefficients that are the result of a wavelet transform. These coefficients thus make for good features for the EEG signal.

## 3.5 Classification

The last step in a BCI system is the training and usage of a classifier. A classifier is a function that, given a sequence of EEG data features, returns

[8] Source: `http://upload.wikimedia.org/wikipedia/commons/2/23/Wavelet_-_Morlet.png`
Last access: 15-06-2014

distinctive values for each class of mental tasks we want to distinguish. For a proper working and accuracy, a classifier needs to be trained with feature vectors of epochs that are related to each class. In this section we discuss the most important classification techniques used in BCI systems.

### 3.5.1 Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) is as the name suggests a linear classifier. This means that this classification technique uses a linear combination of the features present in a given feature vector to determine the class of its source. In order to construct the linear classifier a training data set is needed. For usage in an EEG-based BCI system, this training data is the EEG data resulting from the calibration data that is labeled according to the mental tasks that were performed. The different mental tasks performed form the set of classes that LDA will distinguish. LDA has proven to be a classification technique that is fairly accurate for EEG-based BCIs.
When classifying data, LDA use a linear combination of the features of different observations to group them into mutually exclusive groups according to their related class. The idea is to find the linear combination that projects the values of an input vector $\vec{x}$, containing the features of different observations, on a vector $\vec{w}$ in such a way that the projections can be divided by a hyperplane into groups that only contain projections of observations of the same class. This hyperplane is perpendicular to the vector $\vec{w}$ and has an offset $c$ relative to the origin. For two classes LDA classifies an EEG sequence into class 1 if $\vec{w} \cdot \vec{x} > c$ and into class 2 if not.
For LDA to work it makes two assumptions. The first one being that the probability density functions for observations of each class are normally distributed. The second assumption is that the covariances of each class are equal to each other. This last assumption is also known as the simplifying homoscedasticity assumption.

### 3.5.2 Support Vector Machines

Support Vector Machines, or SVM, is another machine learning technique that can be used for the classification of EEG data. The idea behind SVM is very similar to the one of LDA. Just like LDA, SVM tries to find an optimal hyperplane that divides the observations in different classes based on the features of these observations. The difference between LDA and SVM lies in the fact that LDA defines an optimal hyperplane based on the normal distributions of observations of the same class, whereas SVM only considers the observations closest to the hyperplane. These observations that lie closest

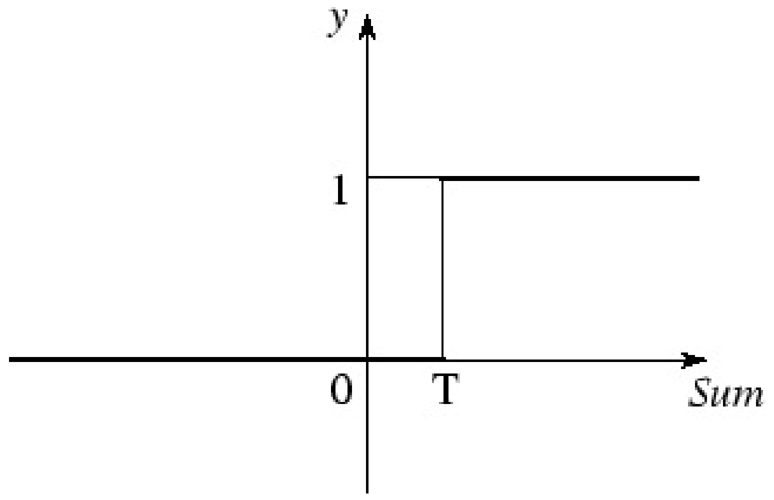


Figure 3.10: Threshold function that returns 1 if the sum of the input values is equal or greater than a threshold value $T$[9]

are called the Support Vectors of the hyperplane since an observation is presented by its feature vector. Hence the name Support Vector Machines. SVM defines the hyperplane that divides the observations of each class in such a way that the margin between the hyperplane and the aforementioned Support Vectors is maximized. The hyperplane is defined by its normal vector $\vec{w}$ and its offset $c$ with respect to the origin, just like with LDA. The problem to be solved is thus very similar to the one in LDA, which is finding $\vec{w}$ and $c$, with the difference of criteria for the optimality of the hyperplane we mentioned of course. To define $\vec{w}$ and $c$ we need a given set of observations as well. As mentioned before, this set of observations consists of the feature vectors resulting from the calibration which are labeled with the performed mental task or in other words class. The resulting classifier is similar to the one of LDA in that an EEG sequence is classified in class 1 if $\vec{w} \cdot \vec{x} > c$ and in class 2 if not, with $\vec{x}$ being the feature vector of the EEG sequence.

### 3.5.3 Artificial Neural Networks

Another machine learning technique that can be used for the classification of EEG signals are Artificial Neural Networks [41]. These Artificial Neural Networks are networks were the nodes represent neurons, like the ones found in humans and animals. Hence the name Artificial Neural Networks. In the

[9] Source: `http://2.bp.blogspot.com/_mtcigb-7B3M/SGZk42FaA5I/AAAAAAAAAHU/f_FdCbVHeWw/s1600/img17.gif`
Last access: 15-06-2014

first chapter we saw the anatomy of neurons and how they communicate with each other. We will give a quick recap here. Neurons receive signals from the synapses of other neurons through their dendrites. These signals accumulate in the body of the neuron. When the accumulated signal in the body of the neuron is strong enough it will trigger a new signal that is being sent across the axon of the neuron. This new signal will then be passed on by the synapses on the axon terminals of the neuron to other neurons. For a representation of a neuron, please refer to Figure 2.1 in chapter 1.
As we already mentioned, the nodes of an Artificial Neural Network try to mimic how actual neurons work. We can state that the nodes in an ANN are actually models of the operation of real neurons. Like real neurons, the nodes of an ANN receive inputs from other nodes or neurons. This means that for each node some inputs are more important than others and thus taken more into account. We can represent the inputs of a node by a vector $\vec{x}$ and their respective weights by a vector $\vec{w}$. Remember that a real neuron accumulates its inputs until it reaches a certain threshold and fires a signal of its own. Since the nodes in an ANN mimic real nodes, they too will accumulate their incoming signals and output a new signal when the accumulation of their inputs reaches a certain threshold. This accumulation is simply a sum of the inputs of the node multiplied by their respective weights. For convenience, lets call this function A. In that case the accumulation in a certain node is represented by equation 3.3 with $j$ being the number of the node.

$$A_j(\vec{x}, \vec{w}) = \sum_{i=0}^{n} x_i w_{ji} \tag{3.3}$$

Notice that $w$ has an index $j$ as well due to the fact that each node has an own set of weights $\vec{w}$ for its inputs $\vec{x}$. Besides this sum function for the accumulation of the input values, a threshold function is needed as well. Such a threshold function returns 0 for values below a certain threshold value $T$ and for instance 1 for values equal or greater than the threshold $T$. Figure 3.10 shows a threshold function that returns 1 if the sum of the input values of a node reaches a threshold value $T$ and 0 if not. The threshold function thus determines the eventual output of node. This output, along with the outputs of other nodes, is then passed on to other nodes, serving as the input of these nodes. These connections between the nodes create a network between them. A Neural Network typically has input nodes and output nodes. Input nodes are nodes that receive their input from outside the Neural Network, which means that they do not receive their inputs from other nodes. Similarly, output nodes send their output outside the Neural Network and therefore not to other nodes. This means that the output of these output nodes gives the result of the decision the Neural Network has made based upon the input data received by the input nodes. However, in order for the Neural Network to make a correct and accurate decision, the Neural Network has to train itself. To train itself a Neural Network uses an algorithm that alters the weights on the inputs of each node to influence how much different parts of the input data affect the decision. Given different sets of input data, called the set of training data, for which the desired output or decision is known, the training algorithm will then make changes to all the weights in the Neural Network until the output or decision for the given input data matches the desired output or decision. The altering of the weights happens in small steps in several recursions to evaluate the effect that each weight has on the output or decision.

# 4 Emotion Recognition

## 4.1 Describing Emotions

To recognise emotions, a distinction between emotions has to be made. The distinction of emotions calls for a description of emotions based on their features in order to compare these features and distinguish them from one another. In the literature two main paths are followed to describe emotions, namely a discrete approach and a dimensional approach.

The discrete approach of describing emotions consists of categorical labeling based on discrete basic emotions. The idea of discrete emotions is that there is a small limited set of basic emotions that are discrete and independent to one another. More complex emotions can be described by a combination of these discrete basic emotions. The foundations of discrete basic emotions was layed by Descartes, who introduced the idea of describing human emotional behaviour based on a few underlying basic emotions. Following this path, Darwin described emotions by distinctive behaviour patterns and stated that these behaviour patterns were selected by evolution due to their survival value[9].
Many different sets of discrete basic emotions and combinations of discrete basic emotions have been proposed over time. However, one of the most used approaches to discrete emotions is the one proposed by Ekman[11]. Ekman

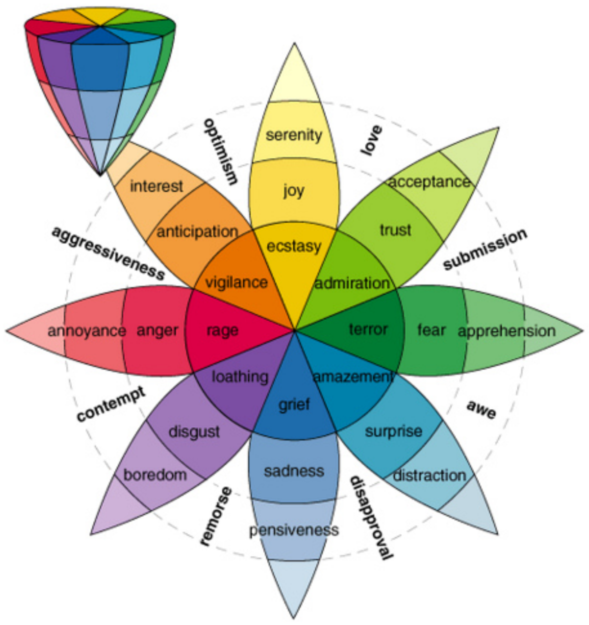


Figure 4.1: 2D and 3D presentation of Plutchik's Wheel[1]

concluded that there are 6 basic emotions that are able to describe facial expressions across different cultures. The 6 basic emotions are anger, disgust, fear, happiness, sadness and surprise.
Another way of describing emotions is proposed by Plutchik[32]. Plutchik uses a set of 8 basic emotions of which 4 emotions are the opposites of the other 4 emotions. The basic opposing emotions are joy versus sadness, surprise versus anticipation, angers versus fear, and trust versus disgust. Plutchik designed a wheel of emotions, as indicated in Figure 4.1, in a 2D and 3D presentation to represent how other emotions can be derived from the 8 basic emotions. For each of the 8 basic emotions, Plutchik defined a low and high intensity which resulted in 16 additional emotions. Furthermore, the 8 basic emotions could be combined side by side to create 8 new emotions, as seen in Figure 4.1. The 8 combinations that result from side by side combination of the basic emotions are optimism, live, submission, awe, disapproval, remorse, contempt and aggressiveness.

Another way to describe emotions is with the dimensional approach. In the dimensional approach emotions are described in a number of dimensions.

[1] Source: `http://trudyvandenberg.files.wordpress.com/2011/05/20110525-081244.jpg?w=640`
Last access: 15-06-2014

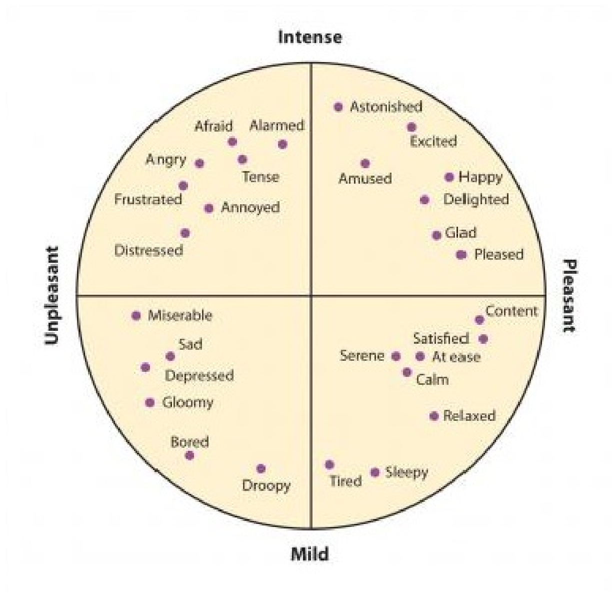


Figure 4.2: Plotting the valence and intensity of emotions for the circumplex model[2]

Different emotions can then be plotted based on their score in each dimension. Two dimensions that are often used are valence and arousal. Arousal is also referred to as intensity or activation. The valence of an emotion can be negative or positive, based on a person's evaluation of events, people or things [8]. The arousal of an emotion can be low or high, based on the strength of the disposition of a person, evoked by the emotion, whether or not to take action [8].

A two-dimensional model with the valence dimension and the intensity dimension is the circumplex model. The circumplex model was introduced by Russel [34]. The circumplex model consists of a two-dimensional circular space with the valence dimension represented on the x-axis and the arousal dimension represented on the y-axis and can be seen in Figure 4.2. Values for valence to the left of the origin represent a negative value and values for valence to the right of the origin represent positive valency. For arousal, values below the origin represent a low arousal and values above the origin represent a high arousal. Emotions are projected in the two-dimensional circular space based on their valence and arousal ratings.
Mehrabian intoduced the PAD emotional state model that extended the cir-

[2] Source: `http://www.peoi.net/Courses/Coursesen/psy3/Resources/fwk-stangor-fig10_003.jpg`
Last access: 15-06-2014

cumplex model with a third dimension named dominance [24]. The dominance of an emotion reflects whether a person feels controlled, or submissive, or rather in control, or dominant.
Plutchik's wheel of emotions can be seen as a hybrid between the discrete approach and the dimensional approach due to the dimension of intensity that is integrated in the model.

## 4.2 Emotions in EEG Signals

To recognise emotions in EEG signals, proper stimuli need to be used to evoke the emotions in the brain. These stimuli can be visual, audio-visual and auditory. In the previous section we discussed how emotion can be classified in theoretical models that follow a discrete approach or a dimensional approach. A lot of researchers that try to detect emotions in EEG signals seem to prefer the dimensional approach over the discrete approach. However, the recognition of discrete emotions in EEG signals has been done as well [26].
As we mentioned in the previous section, two dimensions that often return in the dimensional approach of describing emotions are the dimensions valence and arousal.
Li and Lu [19] introduced an offline EEG-based BCI to detect only the valence of emotions, evoked by pictures, by using an SVM classifier on EEG signals that were band-passed for the gamma frequency band.
Another offline EEG-based BCI was used to detect only the valence of emotions, this time evoked by audiovisual stimuli in the form of short video-clips that were extracted from movies [27]. This last BCI split the EEG signal into the alpha, beta, gamma, delta and theta band and ran an SVM classifier on the Logarithmic Band Power of each frequency band and an additional SVM classifier on the Logarithmic Band Power of all frequency bands. The Logarithmic Band Power is a variation on the Band Power feature extraction where the logarithm of the Band Power is taken to lower the skewness of the distribution of the Band Powers.
Liu et al [21] use the alpha, beta, gamma, delta and theta frequency bands in an EEG-based BCI to recognise emotions evoked by auditory stimuli in the form of music. The BCI of Liu et al classifies the valence of the evoked emotions as well as the arousal. An interesting fact is that Liu et al explicitly use the AF3 and F4 sensors because of their position above the prefrontal cortex.
Schaaff and Schultz [36] use sensors on position Fp1, Fp2, F7 and F8 to recognise valence and arousal evoked by pictures. Schaaff and Schultz use

two approaches for preprocessing and feature extraction. The first approach uses a band-pass filter of 5 to 40 Hz and the average of adjacent frequency components. The second approach uses the highest amplitude in the alpha band and the overall alpha band power as features. Both approaches use an SVM as classifier.
Other research has shown that an emotion with a negative valence goes along with more activity in the right hemisphere of the prefrontal cortex, while an emotion with a positive valence goes along with more activity in the left hemisphere of the prefrontal cortex [37][2]. The research also states that emotions with a high arousal evoke more activity in both hemispheres of the prefrontal cortex. Activity in brain areas goes along with low alpha activity and high beta activity in the specific brain area, while the deactivition of brain areas is related to high alpha activity and low beta activity. Based on the effects of valence and arousal of emotions in the brain, alpha and beta activity in the prefrontal cortex can be used to detect the valence and the arousal of emotions [2]. Furthermore, the use of only the alpha and beta frequency band to detect emotions automatically attenuates artefacts like eye movement and eye blinks, $<$4Hz, ECG or heart artefacts, 1.2Hz, and EMG or muscle artefact, $>$30Hz [2].

# 5 Emotional Browsing

The concept of emotional browsing is browsing through content that is labeled with emotions. In order to be able to browse content based on emotions, the content has to be labeled in the first place. Labeling content with emotions can already be seen in several applications. Like and dislike buttons on several media sharing platforms come to mind where a basic distinction between positive or negative emotions is made. Several other applications include labeling content by adding tags to enable other users to retrieve the content based on these tags. In most cases, the browsing of content is based on the feelings of others toward this content. As emotions are very personal, browsing content based on a users own emotions is preferable. However, actively labeling content by emotion can be a lot of work, which is why a lot of users will not even bother. Emotional recognition through BCIs can solve this. The emotions recognised by a BCI can be used to label content while the user is just watching the content. Therefore, a BCI that recognises emotions does not require the user to do anything to label content, besides wearing an EEG capture device. Since it is not unthinkable that EEG capture devices will become smaller and more user friendly in the future, BCI-based emotional browsing is a promising solution for the shortcomings of current emotional browsing strategies.

In Section 5.1 of this chapter we present our framework that enables applications to integrate the use of an emotion recognising BCI for emotional browsing. Also, we describe an application that we built on top of our framework in Section 5.2. The application is named Emages and uses our framework to passively label images with emotions recognised by the BCI in our framework.

## 5.1 Framework for Emotional Browsing

BCIs are mostly written for the use with a single specific application. These applications are written around their BCI in such a way that the application can not be seperated from the BCI. Because of the lack of generic BCI solutions, the development of an application that is BCI-driven entails writing an own BCI. However, some developers might not be familiar with the field of BCIs. To enable developers to use an emotion recognising BCI in their application without having to know any details of the working of the BCI, we created a framework. The framework allows applications to use a real-time detection of four emotions which are happy, angry, sad and calm. The real-time detection of the emotions implies the use of an online BCI in our framework.

The framework itself consists of an actual EEG-based emotion recognising BCI that we created and a layer on top of the BCI to enable communication with the BCI. The communication layer can be used by an application to start or stop the BCI process, to send messages to the BCI and to receive classification results of the BCI process. Because a BCI uses machine learning algorithms that need to be trained in order to classify brain signals, the application needs to implement a training session. This training session simply consists of showing a small set of pre-labeled content for each of the four emotions. It is the responsibility of the application use the framework to notify the BCI of the labels of the pre-labeled content. This explains why the framework supports sending messages to the BCI.

We also provided an easy way for developers that are familiar with EEG-based emotion recognising BCIs to modify the design of the BCI embedded in the framework. This can be done by changing the configuration of the framework in order to load a different OpenVibe scenario. OpenVibe is an environment to design BCIs. OpenVibe and our BCI design are further discussed in Chapter 6.

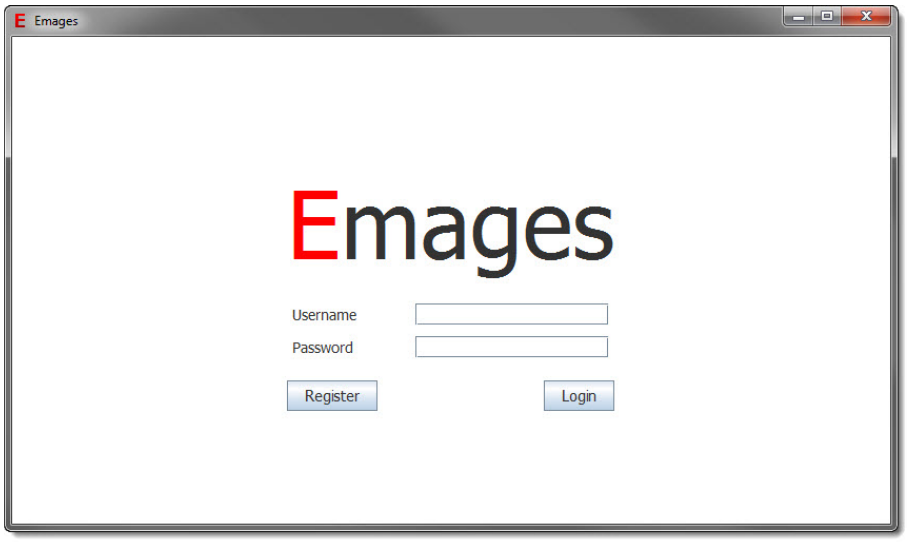


Figure 5.1: Screen to log in into the application

## 5.2 Emages Application

The Emages application is built upon our framework and allows users to browse through images while automatically labeling them based on the emotions the images evoke in the brain of the user. Later on, the user can then retrieve and view images based on the emotions felt while browsing through the images. In this section we describe how the Emages application looks and can be used. We kept the look and feel of the GUI as simple, clean and intuitive as possible to create a high usability. Note that this section does not contain any details on the implementation of the application. For details on the implementation we refer to Chapter 6.
The first screen encountered in the application is the screen to log in a user into the application and can be seen in Figure 5.1. If a user does not have any login information yet, the register button can be clicked to create a new user. The register button brings a user to the registration screen in Figure 5.2. If a user logs in into the application, his username is looked up into the database and the entered password is compared to the stored password. After a succesful login the main screen in Figure 5.3 is shown. After a user is logged in, his name will appear in the top right corner of every screen along with the option to log out. When a user logs out he is taken back to the login screen.
Figure 5.2 shows the screen used to register a new user. When all information is correctly filled in and submitted, the user will be registered in the system and able to log in and use the application. After registering the user is taken

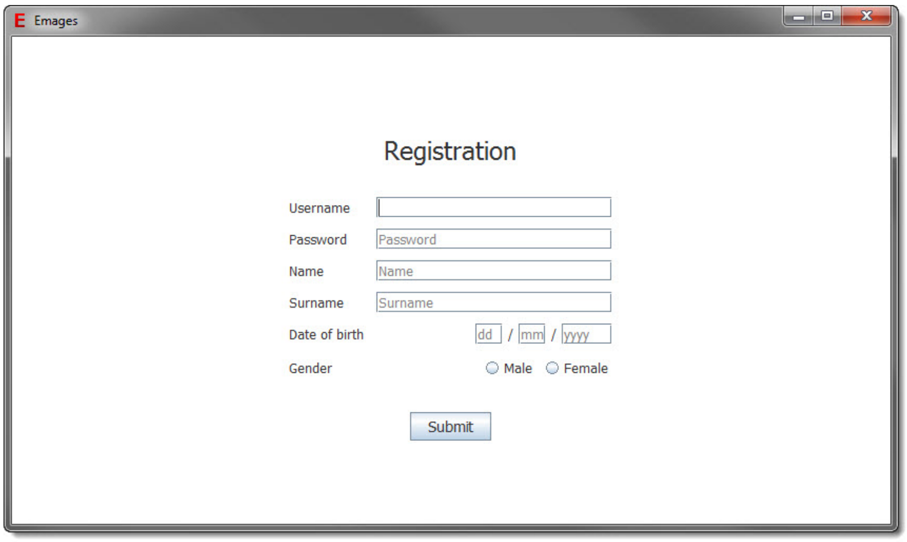


Figure 5.2: Screen to register a new user for the application

back to the login screen.
The main screen is shown in Figure 5.3. The main screen offers the option to browse and classify images, our to retrieve classification results. The button for browsing leads to the screen in Figure 5.4, while the button for retrieving images leads to the screen in Figure 5.9.
When a user wants his emotions to be detected he first needs to train the embedded BCI. To browse images, the first screen encountered is a loading screen, as illustrated in Figure 5.4. This loading screen is shown until the BCI training process is loaded in the background.
When the BCI training process is loaded in the background the screen automatically switches to the screen to start the training, shown in Figure 5.5. This screen only serves as a notification to warn the user that the training will start after this screen. The user can choose to return to the main screen or to start the training. The start button takes the user to the screen shown in Figure 5.6 and starts the BCI training process.
This figure shows the screen for the actual training of the BCI. The screen will play an automatic slideshow of the training images. Each training image is shown for 5 seconds followed by a black screen of 8 seconds. The reason for showing the black screen is to reset the users state of mind. When all the training images are shown the screen automatically switches to the screen in Figure 5.7. At any time of the training process, the user can choose to go back to the main screen by clicking the according button.
Figure 5.7 shows the screen that is used to notify the user that the training of

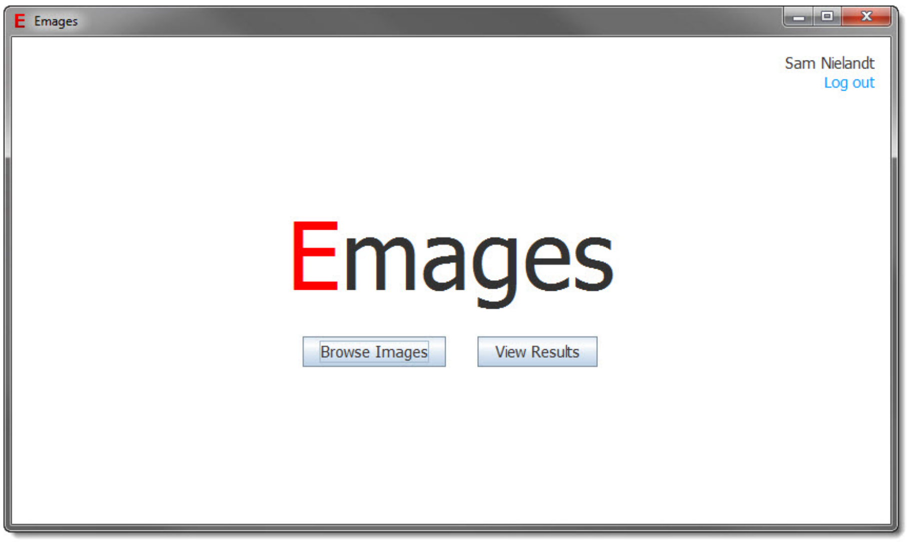
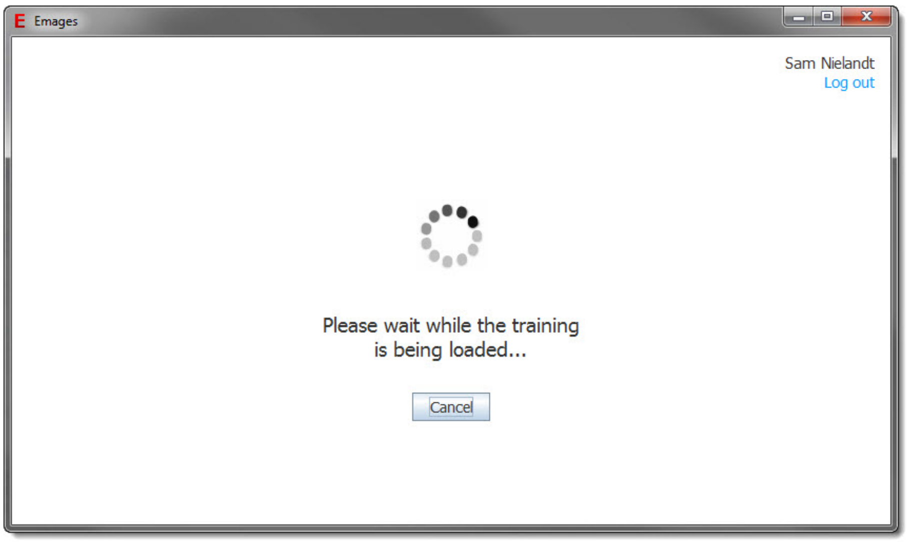


Figure 5.3: Screen with the main menu of the application



Please wait while the training
is being loaded...

Figure 5.4: Screen to begin the training process

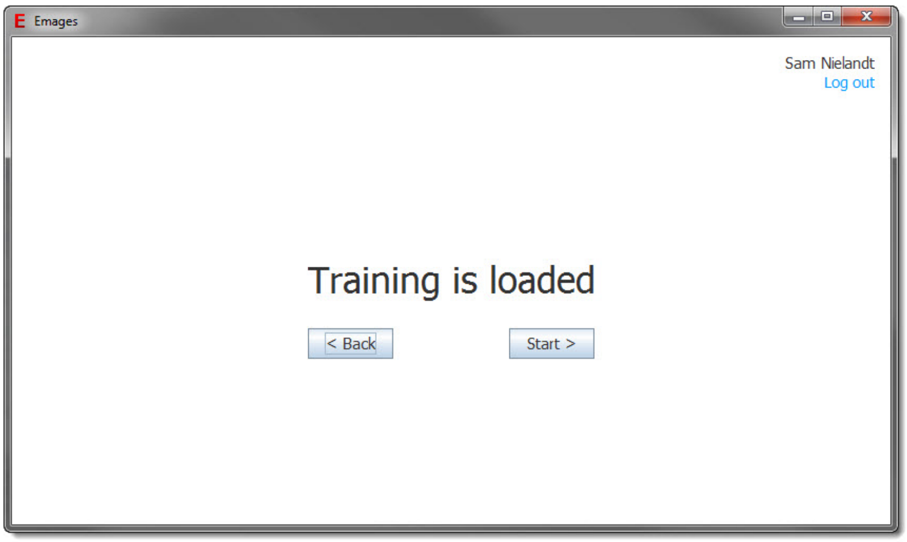


Figure 5.5: Screen to begin the training process

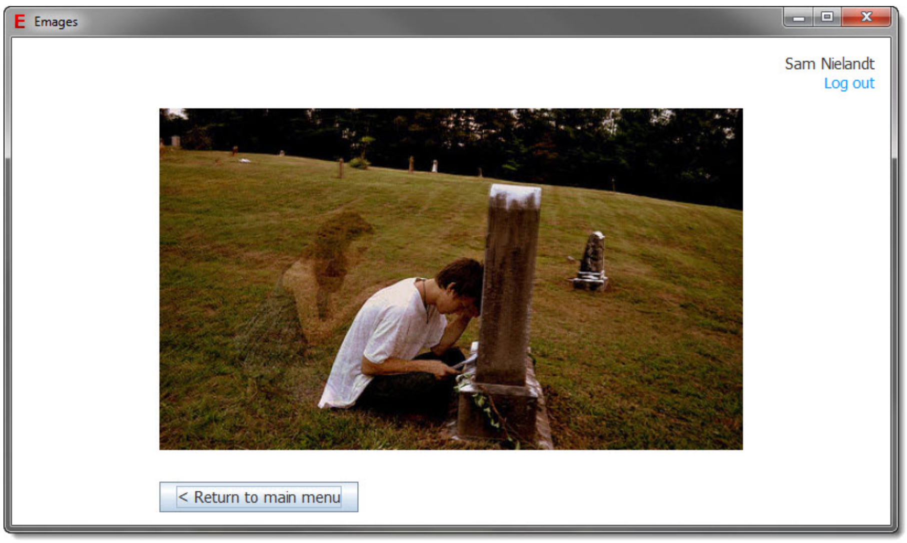
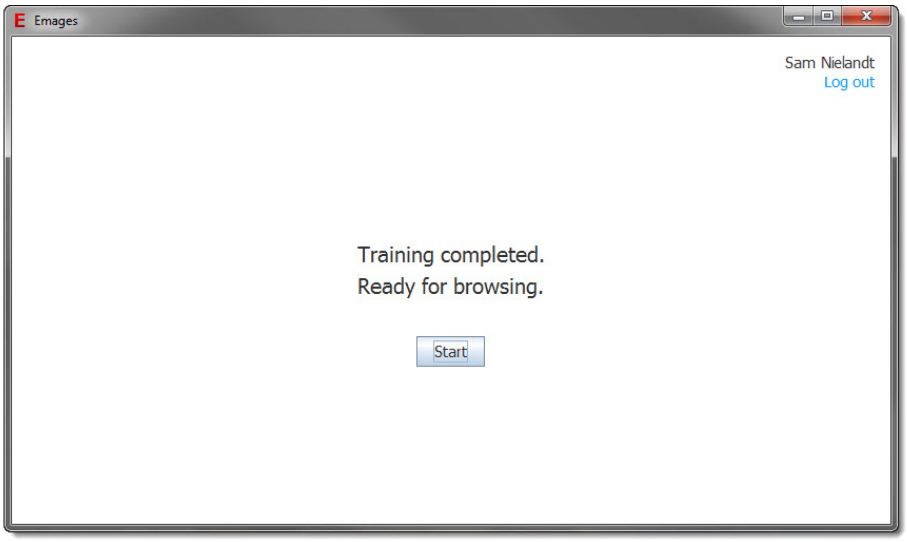


Figure 5.6: Screen containing a slideshow of the images for the training process. Training signals are gathered while the user views the images.



Training completed.
Ready for browsing.

Figure 5.7: Screen to indicate that the training process is done and that the user can begin browsing

the classifiers is completed and to start the actual browsing of images. When clicking the start button the screen switches to the screen in Figure 5.8 and the BCI process for recognising emotions is started.
The screen in Figure 5.8 is used to browse through images. The look of the screen is the same as the screen for the BCI training process. Just like the screen for the training process, a slideshow of images is played in which each image is shown for 5 seconds, followed by a black screen that is showed for 8 seconds. The user decides when he stops browsing by clicking the button to return to the main screen and end the BCI process. After an image is watched, the classification result of the BCI for the image is stored in the system.
The screen in Figure 5.9 allows the user to retrieve the images for which the emotion of the user was detected while browsing. Based upon the filters selected by the user, the images that are labeled with the related emotion are shown. When the user clicks on a miniature he can view the large version of the image along with details on the classification of the image in the screen in Figure 5.10.
The screen in Figure 5.10 shows an image and its classification result. The user can return to the retrieved results by clicking the button.

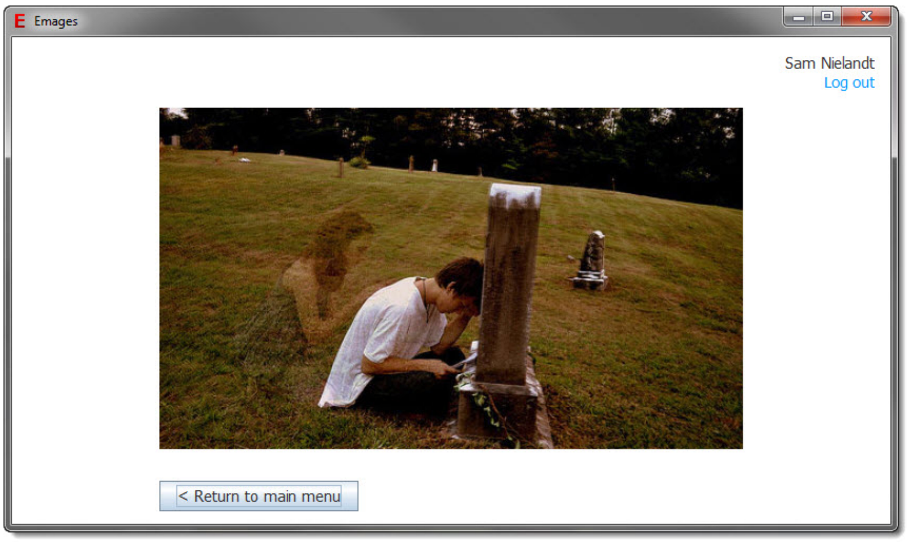


Figure 5.8: Screen containing a slideshow of the images for the browsing process. The images are being classified while the user views the images.

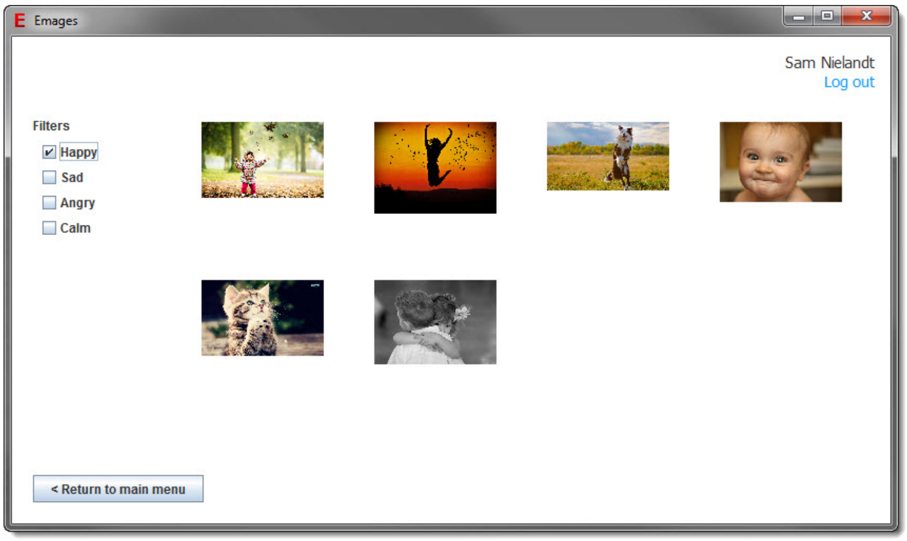
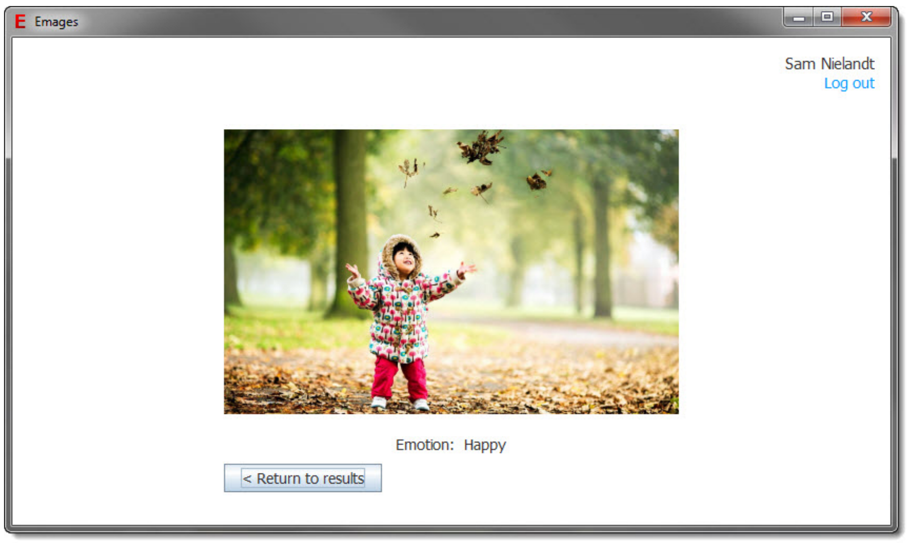


Figure 5.9: Screen to retrieve images based on emotions



Emotion: Happy

Figure 5.10: Screen to view the details and classification results of an image retrieved by emotion

# 6 Implementation

We investigated the structure of BCIs and the used techniques in BCIs as well as theoritical models of emotions and how emotions can be detected in EEG signals. Based on our research in the literature, we were able to make well-educated decisions for the implementation of our framework and the Emages application we proposed in this thesis. In this chapter we discuss the implementation of our framework and the Emages application in detail.

## 6.1 Overview

To use the Emages application, a user needs to wear an EEG capture device to measure EEG signals generated by the users brain. To measure the EEG signals we used the Emotiv Epoc, which is discussed in Section 6.2 of this chapter. An overview of the structure of the application is presented in Figure 6.1.
The Emages application is implemented in two components, a Java application component and an OpenVibe component. OpenVibe[1] is an environment to design and run BCI. Towards this end, OpenVibe has a designer application to design and execute BCIs, and an Acquisition Server to receive EEG signals from an EEG capture device and send the EEG signal to the Open-

[1] OpenVibe website: `http://openvibe.inria.fr/`

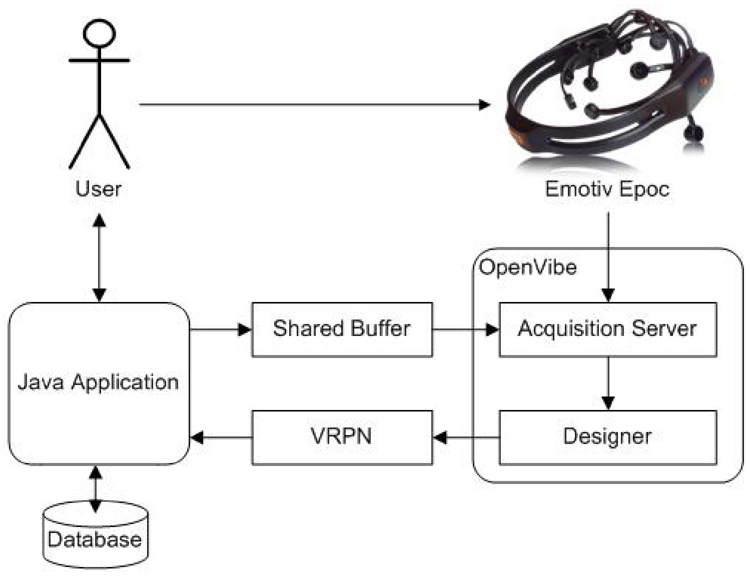
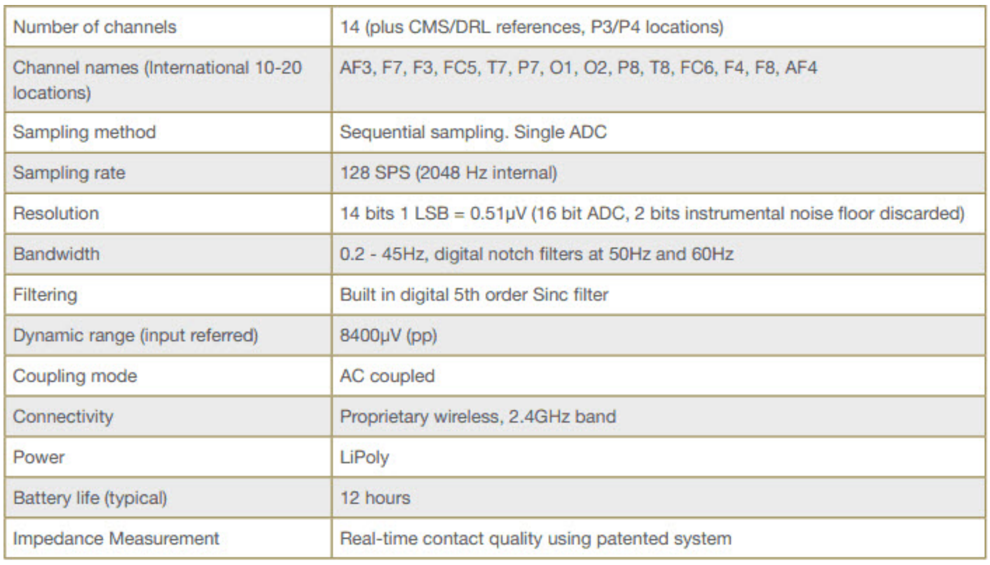


Figure 6.1: Overview of the structure of the application

Vibe designer. OpenVibe features a modular approach towards the design of Brain Computer interfaces. This means that we can choose functional modules and configure and connect them to build the BCI and the trainer for the BCI.
As we mentioned in Section 5.1 of Chapter 5, we aimed for an online BCI. An online BCI enables applications that are built upon our framework to recognise emotions in real-time, ensuring a wider range of possibilities than an offline BCI would offer. The downside of an online BCI is that certain preprocessing techniques like ICA or manual artefact detection are impossible. This causes the BCI to be more sensitive to artefacts and might result in a lower classification accuracy.
The main component in our system is written in Java. The Java component controls the storage and retrieval of data a MySQL[2] databases. The details on the database and the access to it are described in Subsection 6.6.3. Furthermore, the Java component controls when a BCI is started by OpenVibe in the background. However, the application only starts the OpenVibe designer. We deliberately did not embed the OpenVibe acquisition server in the framework to provide the user the possibility to work with different EEG capture devices. Details on how to configure and use the OpenVibe acquisition server can be found in Subsection 6.3.1. The Java application

[2] MySQL website: `http://www.mysql.com/`



| | |
|---|---|
| Number of channels | 14 (plus CMS/DRL references, P3/P4 locations) |
| Channel names (International 10-20 locations) | AF3, F7, F3, FC5, T7, P7, O1, O2, P8, T8, FC6, F4, F8, AF4 |
| Sampling method | Sequential sampling. Single ADC |
| Sampling rate | 128 SPS (2048 Hz internal) |
| Resolution | 14 bits 1 LSB = 0.51μV (16 bit ADC, 2 bits instrumental noise floor discarded) |
| Bandwidth | 0.2 - 45Hz, digital notch filters at 50Hz and 60Hz |
| Filtering | Built in digital 5th order Sinc filter |
| Dynamic range (input referred) | 8400μV (pp) |
| Coupling mode | AC coupled |
| Connectivity | Proprietary wireless, 2.4GHz band |
| Power | LiPoly |
| Battery life (typical) | 12 hours |
| Impedance Measurement | Real-time contact quality using patented system |

Figure 6.2: The Emotiv Epoc specifications[3]

contains the logic to drive the BCI training process and to interpret classification results of the BCI as well by communicating with OpenVibe. This communication happens by sending and receiving labels to and from OpenVibe and receiving classification values from OpenVibe. Sending labels to OpenVibe happens through the use of a shared buffer. More details on the implementation and the use of the shared buffer are provided in Section 6.4. To receive values and labels from OpenVibe, VRPN is used, which stands for Virtual Reality Peripheral Network. In Section 6.5 we explain what VRPN is and how we used it in our implementation. Finally, the Java application has a GUI to allow the user to interact with the Emages application and to provide feedback to the user.

## 6.2 Emotiv Epoc

To capture the EEG signals we used the Emotiv Epoc EEG capture device, which is depicted in Figure 3.1. Emotiv[4] is a neuroengineering company that offers wireless EEG capture solutions along with different SDK versions to develop BCI-driven applications. We used the Research Edition of the Emotiv Epoc that comes with the Research SDK version. The Emotiv SDK offers

[3] Source: `emotiv.com/upload/manual/EPOCSpecifications.pdf`
Last access: 15-06-2014

[4] Emotiv website: `http://emotiv.com/`

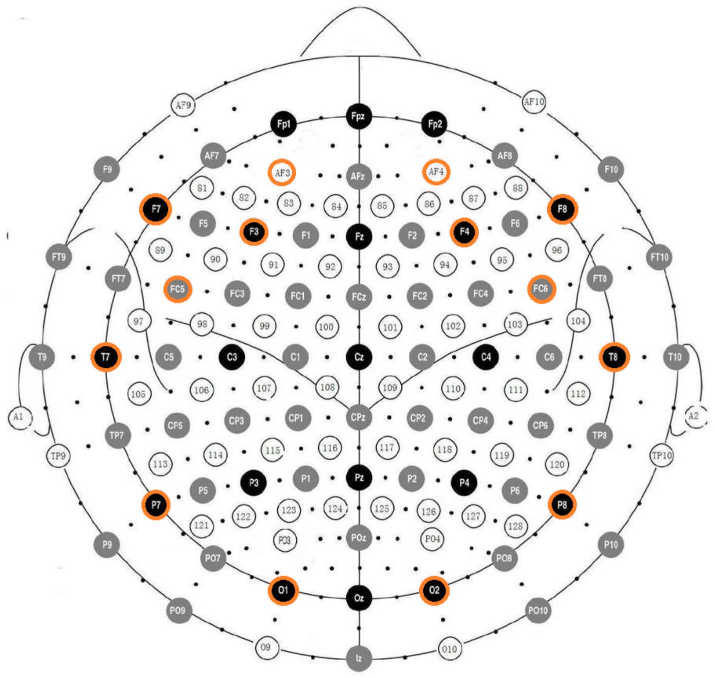
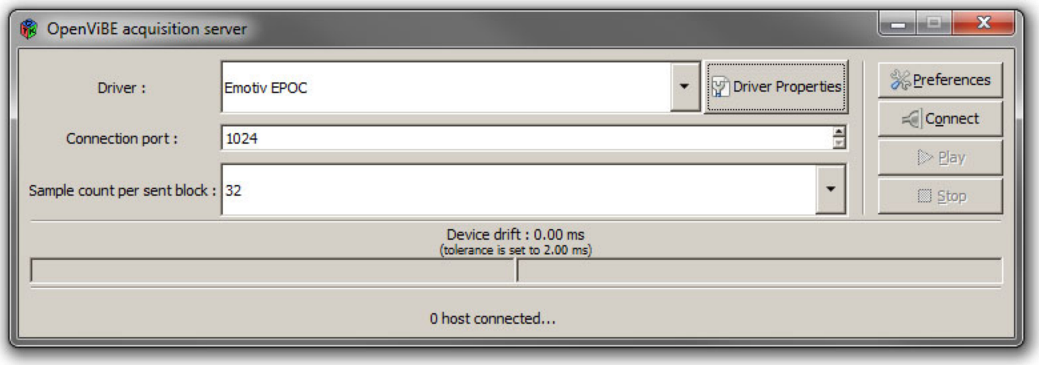


Figure 6.3: The locations of the electrodes of the Emotiv Epoc highlighted in orange according to the international 10-20 localisation system[5]

out of the box solutions for acquiring BCI classification results. However, no access to the source code or the use of specific techniques and algorithms for the actual BCI process are given, nor are the used techniques and algorithms for the BCI process in the Emotiv SDK described. Because of the limitation in access and knowledge of underlying techniques and algortihms we do not have control over the actual BCI process and we can not describe how the BCI process was implemented. Because of the limitation in control and knowledge of the BCI process we opted to implement our own BCI in OpenVibe while still using the Emotiv Epoc as the EEG capture device. The biggest advantage of the Research SDK version in our case is that the Research SDK version gains access to the driver for the Emotiv Epoc. This driver allows us to use the Emotiv Epoc device together with OpenVibe. Details of how to connect the Emotiv Epoc capture device are describe in Subsection 6.3.1 on the use of the OpenVibe Acquisition Server.
The Emotiv Epoc device is an EEG capture device that connects wirelessly

[5] Source: `http://img3.wikia.nocookie.net/__cb20091213024356/emotiv/images/thumb/4/43/Emotiv_elec_layout.png/500px-Emotiv_elec_layout.png`
Last access: 15-06-2014


OpenViBE acquisition server

Driver : Emotiv EPOC

Connection port : 1024

Sample count per sent block : 32

Device drift : 0.00 ms
(tolerance is set to 2.00 ms)

0 host connected...

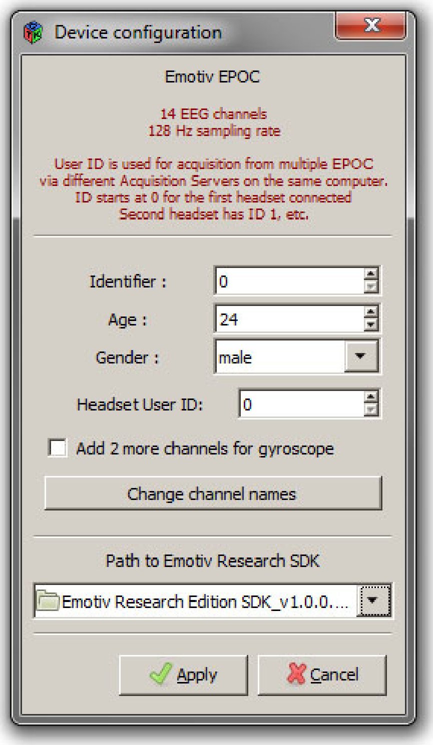
Figure 6.4: Screenshot of the OpenVibe Acquisition Server

to a machine that is responsible for the EEG signal acquisition. This machine receives the EEG signals from the Emotiv Epoc through a USB dongle. Figure 6.2 shows the specifications of the Emotiv Epoc device. The specifications of the Emotiv Epoc state that the Emotiv Epoc has a sampling rate of 128Hz which means that it measures 128 samples of the EEG signal per second. Furthermore, the Emotiv Epoc device has built-in notch filters at 50Hz and 60Hz to remove line noise as an artefact.
The Emotiv Epoc device has 14 electrodes to measure EEG signals and 2 electrodes that serve as references to the 14 electrodes measuring EEG signals. In the design of the Emotiv Epoc device, the 14 electrodes have a fixed location according to the international 10-20 localisation system, described in Subsection 3.2.1 on EEG. The locations of the 14 electrodes measuring EEG signals are AF3, AF4, F7, F8, F3, F4, FC5, FC6, T7, T8, P7, P9, O1 and O1, and are highlighted in orange on an image representing the international 10-20 localisation system in Figure 6.3. The 2 reference electrodes are placed on locations P3 and P4. Apart from the electrode channels, the Emotiv Epoc also uses a two axis gyroscope that measures horizontal and vertical movement of the head.

## 6.3 OpenVibe

### 6.3.1 Acquisition Server

The OpenVibe acquisition server can be started separately from the OpenVibe designer. Upon start-up a screen is shown that can be seen in Figure 6.4. In this screen the EEG capture device to be used can be chosen. Since we will be using the Emotiv Epoc as capture device we will choose this device


Device configuration

Emotiv EPOC

14 EEG channels
128 Hz sampling rate

User ID is used for acquisition from multiple EPOC
via different Acquisition Servers on the same computer.
ID starts at 0 for the first headset connected
Second headset has ID 1, etc.

Identifier : 0
Age : 24
Gender : male
Headset User ID: 0

☐ Add 2 more channels for gyroscope

Change channel names

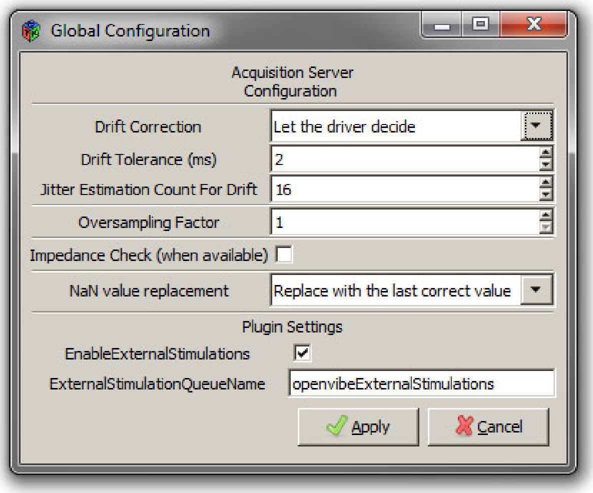
Path to Emotiv Research SDK
Emotiv Research Edition SDK_v1.0.0....

Apply Cancel

Figure 6.5: Screenshot of the configuration screen of the chosen EEG capture device for the OpenVibe acquisition server. This is the configuration for the Emotiv Epoc capture device

from the list. Furthermore, the connection port used between acquisition server and client can be specified. This connection port is 1024 by default. We kept the default value of 1024 for the acquisition server and client. It is important that the connection port matches the connection port specified in the configuration of the acquistion client in the OpenVibe designer. We will go into further detail on the acquisition client in subsections 6.3.3 and 6.3.4. The last input box of the OpenVibe acquisition server screen lets you specify sample count per block. The acquisition server sends the received data from the EEG capture device to the acquisition client in blocks of data. The sample count per block, as the name suggests, lets you specify how much samples from the measured EEG signal are sent in one block. Basically this means that you can control the size of the blocks. We left the sample count per block at value 32, which is the default value.
Before running the acquisition server we should make sure that the EEG capture device is configured properly to be able to connect with the acquisition server and to work as expected. To configure the EEG capture device a but-

ton that says "Driver Properties" can be clicked. This opens up a new screen that can be viewed in Figure 6.5. Note that the available configuration in this screen depends on the chosen EEG capture device. Figure 6.5 presents the specific configuration screen for the Emotiv Epoc.
The most important part of this configuration screen is to specify the location of the Emotiv Epoc driver. As we already mentioned in Section 6.2, this driver is only available in the research edition of the Emotiv SDK. Without this driver and without specifying its correct location in the configuration screen, the OpenVibe acquisition server will not be able to communicate with the Emotiv Epoc. Besides specifying the location of the driver for the capture device the screen lets you specify information about the user like an identifier, age and gender. This information serves as experiment information. We do not use this information in OpenVibe so it really does not matter what is filled in here. Instead we keep track of the identity of a user in the Java application section for the purpose of storing data linked to the user in a database. Apart from specifying user information in the device configuration for the OpenVibe acquisition server, an identifier can be assigned to the EEG capture device itself as well in case you would use two EEG capture devices. Another option in the configuration screen is to add the 2 channels for the gyroscope of the Emotiv Epoc. In our proof of concept, the gyroscopes are not used. The last configuration option in the configuration screen is to rename the channel names of the EEG capture device. However, the OpenVibe acquisition server is by default configured to use the 10-20 localisation for a correct mapping of the right names to the right channels. It is therefore a good idea to leave the mapping of the names to the channels to the acquisition server.
The last step before running the acquisition server is the configuration of the acquisition server itself. In the screen presented in Figure 6.4 a button that says "Preferences" will open the configuration screen for the server. This screen can be seen in Figure 6.6. A first configuring option are the settings for the drift correction. Drift happens when some samples of the EEG signal are suddenly missing or duplicated when they are being sent from the EEG capture device to a system. Missing samples or additional samples can cause new incoming samples to be mapped wrong on the time axis. The samples following are therefore mapped to a point in time that does not match the actual time that they were measured. Because the interpretation of EEG signals is very time sensitive, drifts in an EEG signal that become too big can cause the BCI to return inaccurate results. To correct drifts in the EEG signal we chose the option "Let driver decide", meaning that drift correction from the driver for the Emotiv Epoc will be used. Another important setting



Acquisition Server Configuration

| | |
|---|---|
| Drift Correction | Let the driver decide |
| Drift Tolerance (ms) | 2 |
| Jitter Estimation Count For Drift | 16 |
| Oversampling Factor | 1 |
| Impedance Check (when available) | ☐ |
| NaN value replacement | Replace with the last correct value |

Plugin Settings

| | |
|---|---|
| EnableExternalStimulations | ✓ |
| ExternalStimulationQueueName | openvibeExternalStimulations |

Figure 6.6: Screenshot of the configuration screen of the OpenVibe acquisition server. This is the configuration for the acquisition server itself

in the acquisition server configuration screen for our application is the setting to enable external stimulations. As discussed in Section 6.1 the Java application will be sending labels to OpenVibe. In OpenVibe, labels are called stimulations because they are not only used to label the EEG signal, but they can also be used to indicate certain events that are not related to the EEG signal. Labels sent from the Java application, or any other external application for that matter, are being picked up by the acquisition server through the use of a shared buffer. The name of this shared buffer can be changed in the configuration screen for the acquisition server. Note that this name is used in the Java application as well to identify the shared buffer, meaning that the names provided in the acquisition server configuration and the Java application should match at all times. Upon receiving labels through the shared buffer the acquisition server will pin them on the EEG signal and send the labels along with the EEG signal to the acquisition client.

### 6.3.2 Designer

In OpenVibe the actual design of a BCI is done in the OpenVibe designer, as the name might suggest. A screenshot of the designer application can be seen in Figure 6.7. We already mentioned that implementing a BCI in OpenVibe is done by connecting modules. These modules are called boxes in OpenVibe. The available boxes are organised in the right column of the designer according to the step in the BCI process they belong to or according

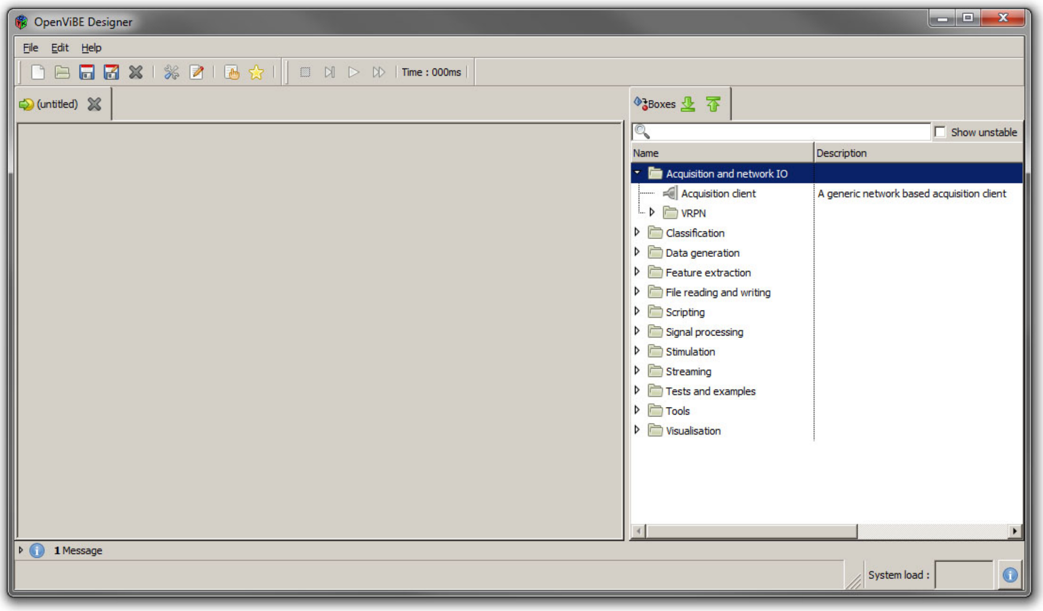


Figure 6.7: Screenshot of the OpenVibe Designer

to their function. Discussing the function and usage of each box is beyond the scope of this thesis. For more information on all available boxes we refer to the box documentation page on the OpenVibe website[6]. However, we will discuss the configuration and usage of the boxes used in the training part of the BCI in Subsection 6.3.3 and in the online part of the BCI in Subsection 6.3.4. To use boxes in a BCI design you can simply drag them from the box browser in the right column of the designer to the grey main panel. A box can have one or more inputs and outputs, depending on the type of box. Each input or output has an assigned data typed making sure that the output from one box can only be connected to an input of another box with a matching or compatible data type.
Designs for a BCI are called scenarios in OpenVibe and can be saved in the xml file format. A BCI in OpenVibe can only be ran inside the designer. To run a BCI in OpenVibe the scenario of the BCI should be opened and the play button in the toolbar should be pressed. The toolbar also offers a pause button to pause the execution of the BCI and a stop button to terminate the BCI process. A BCI can also be started through the command line. To do this a cmd file, provided by OpenVibe, can be opened in the command line along with some optional parameters to open or immediatly play a scenario

[6] OpenVibe Box Documentation: `http://openvibe.inria.fr/documentation/unstable/Doc_BoxAlgorithms.html`

when the designer is opened. Furthermore an additional parameter can be specified to hide the GUI in order to run the BCI in the background. Provided that the xml file of a scenario is located in the same directory as the cmd file the following command can be entered in the command line to immediatly run a BCI design saved as a scenario in the xml file 'Trainer.xml' in the background: 'openvibe-designer.cmd –play Trainer.xml –no-gui'.

### 6.3.3 BCI Trainer Design

When training the classifiers of the BCI, the OpenVibe acquisition server will receive the labels to label the EEG signal through the shared buffer. The acquistion client in the OpenVibe designer will then receive the labeled EEG signal from the acquisition server and the actual training of the BCI is done in OpenVibe. This subsection explains what techniques are used, how these techniques are implemented in OpenVibe and how training part of the BCI is designed as a whole. Figure 6.8 gives an overview of the whole BCI training design in the OpenVibe designer. Because of the large size of the design we will show more detailed images of the design as we explain the design step by step.
In order to detect the emotions we first need a theoretical classification model of emotions for which the classification procedure can be translated to an EEG-based BCI. In Section 4.1 of Chapter 4 on emotions we mentioned that the classification procedure of the circumplex model can be easily translated to an EEG-based BCI. The circumplex model classifies emotions based on their valence and arousal. We also discussed how the difference in positive and negative valence can be detected in EEG signals, as well as the difference between high and low intensity. To summarise, for a negative valence high alpha and low beta activity is measured in the prefrontal cortex in the left hemisphere, and for a negative valence high alpha and low beta activity is measured in the prefrontal cortex in the right hemisphere. For low arousal higher alpha activity and lower beta activity is measured in the prefrontal cortex for both hemispheres, and for high arousal lower alpha activity and higher beta activity are measured in the prefrontal cortex for both hemispheres. Furthermore, the electrodes F3 and F4 were shown to provide the most representing EEG signal because they are positioned above the prefrontal cortex.
Figure 6.9 shows the first part of the BCI trainer design in OpenVibe. This part of the BCI trainer design includes the data acquisition and the preprocessing.

For the *data acquisition* we need to obtain the labeled EEG signal from

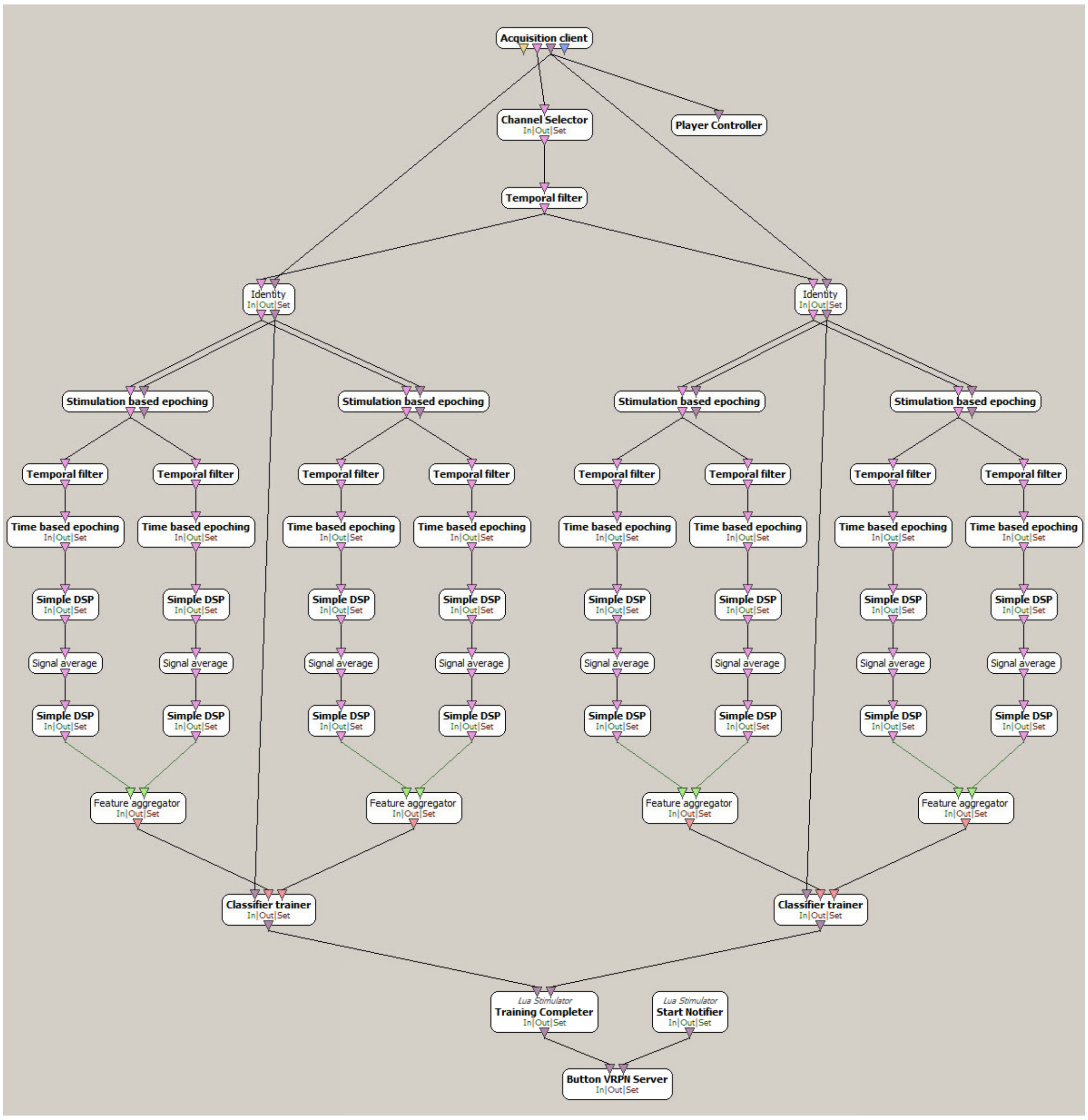


Figure 6.8: Overview of the BCI design for training the classifier

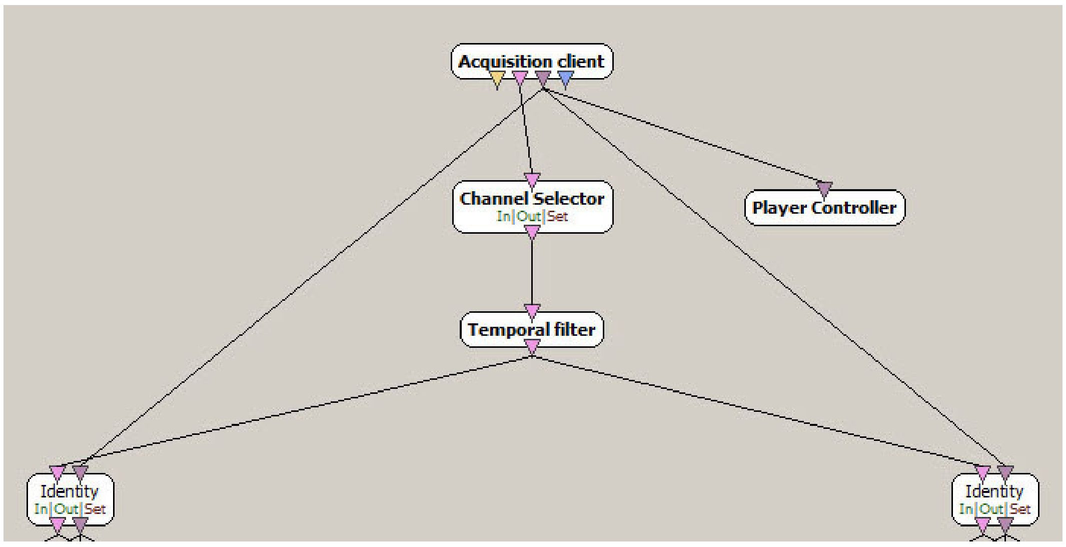


Figure 6.9: Data acquisition and the preprocessing for the BCI training design

the OpenVibe acquisition server. In order to obtain the labeled EEG data we needed the acquisition client box. The client box is configured by double-clicking it which brings up a configuration window. In the configuration the hostname of the machine on which the acquisition server is running needs to be specified. We left the default value which refers to an OpenVibe variable for the hostname which is 'localhost' by default because we ran the acquisition server on the same machine as the BCI. Furthermore the connection port between the acquisition server and the acquisition client can be specified. The acquisition client box outputs the EEG signal, for which the output is represented by a pink triangle, and outputs the labels, for which the output is represented by a purple triangle. These specific color codes are used for these data types throughout OpenVibe. The first box that receives the label stream from the acquisition client is the Player Controller box. The Player Controller box is used to control the execution of the BCI scenario by detecting a trigger label in the label stream to either start or stop playing the BCI scenario. We need the Player Box here in case we want to cancel the training of the BCI in the Java Application. A label can then be sent from the Java application to OpenVibe to stop executing the training design for the BCI.

For the *preprocessing* part we applied the Channel Selector box to only use the EEG signals of the F3 and F4 because these are the ones that are of most interest to detect emotions in EEG signals. By only using the F3 and

F4 channel we greatly increase user-friendliness. The Channel Selector box receives the EEG signal for each channel. In the configuration of the Channel Selector box the names of electrodes or channels, which is 'F3;F4' in our case, must be specified along with the action to either reject or select the channels, which is 'select' in our case. The Channel Selector box then outputs the allowed channels. The next part of the preprocessing step consists of band passing the EEG signal to only let the alpha and beta frequency bands pass. To do this we used the Temporal Filter box. The Temporal Filter box can be configured to be either a low pass, high pass, band stop or band pass filter in its configuration. Because these are all filters that filter the signal in its frequency spectrum we think that the name 'Temporal Filter' for the box is a bit misleading. We chose to configure the Temporal Filter box as a band pass filter. In the configuration of the Temporal Filter box the low cut-off frequency and the high cut-off frequency needs to be specified if the box is being used as a band pass filter. We entered 8Hz for the low cut-off frequency and 30Hz for the high cut-off frequency because we only need the alpha and beta frequency bands to detect emotions, for which the frequency bands are respectively 8Hz-12Hz and 13Hz-30Hz. By only passing the alpha and beta frequency bands we also remove artefacts that operate at frequencies outside the alpha and beta frequency bands. By selecting channels before using the band pass filter the EEG signal is only band passed for two channels instead of 14 channels, which saves computation time. After the channel selection and the band pass filtering, the signal is sent to two boxes called Identity. The label stream from the Acquisition Client passed to the two boxes called Identity as well. The Identity box simply duplicates its inputs on its outputs without performing a function on the input data. The Identity box is therfore simply used for the purpose of layout in the OpenVibe designer and does not server an actual function in the BCI.
Figure 6.10 shows the next steps of the BCI trainer design, which is the feature extraction step and the classifier training step. Notice that Figure 6.10 only shows the feature extraction and classifier training for the distinction between negative and positive valence. Note that the feature extraction step is the same for both the training of the classifier for valence and the classifier for arousal.

The first step in the *feature extraction* part is to extract the epochs that represent each class from the EEG signal and to organise the epochs per class label. The epoching of the EEG signal and dividing the resulting epoch according to class is done by the Stimulation Based Epoching box. This box accepts the EEG signal and the label stream. In the configuration of

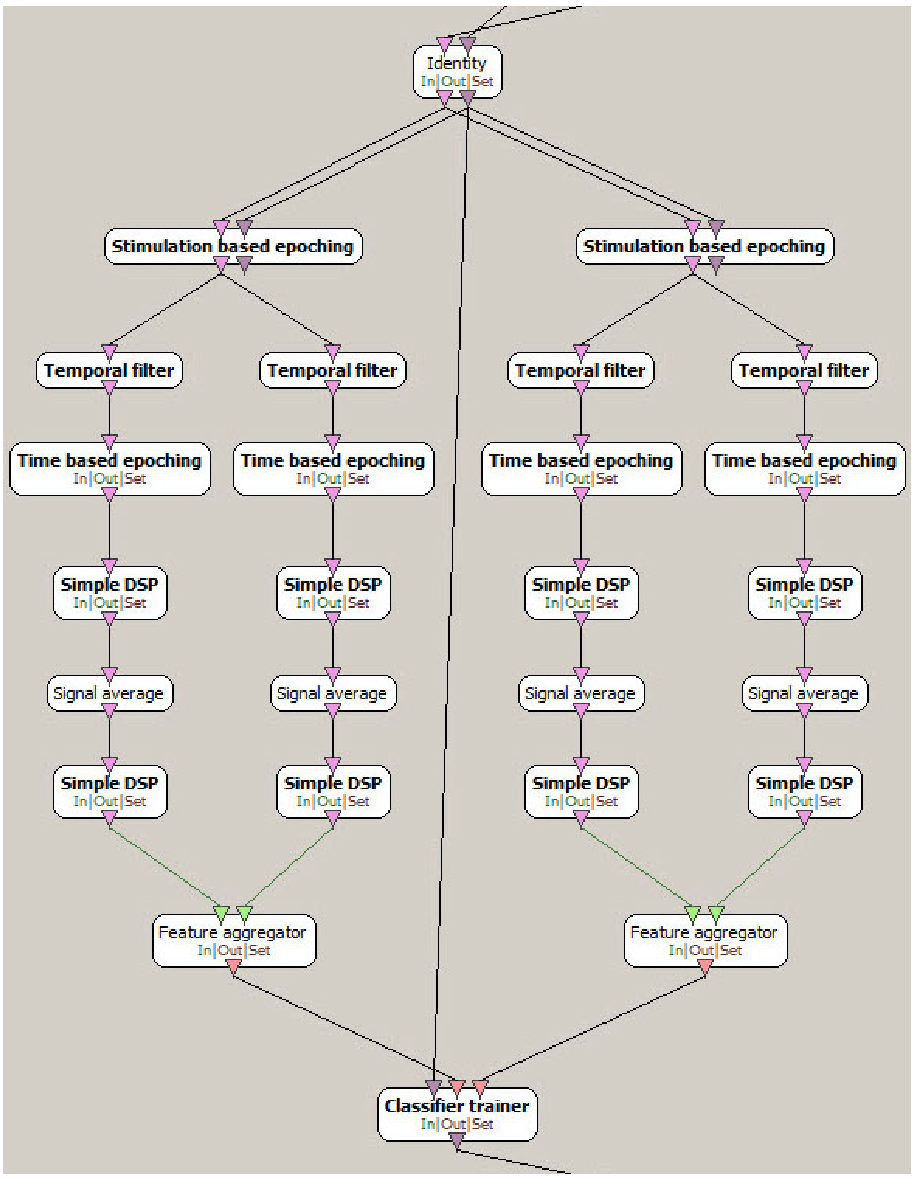


Figure 6.10: The feature extraction and the classifier trainer for the BCI training design

the Stimulation Based Epoching box the label to start an epoch from can be specified as well as the duration in seconds the epoch should have and the offset of the starting point in seconds of the epoch towards the point in time of the trigger class label. We chose a duration of 5 seconds and an no offset. To indicate epoch starting points for the negative valence class we used the label `OVTK_GDF_Left` that is integrated in OpenVibe and the label `OVTK_GDF_Right` for the positive valence class. For the high arousal class we used the label `OVTK_GDF_Up` and for the low arousal class we used the label `OVTK_GDF_Down`. The labels indicating left, right, up and down correspond to the axis representation of the circumplex model shown in Figure 4.2.

The features that are extracted out of the EEG signal in the epochs should be meaningful towards what we want to detect in the EEG signal and should enhance the results of the classifier that operates on them. Remember that for the detection of valence and arousal in the EEG signal we need to make differentations in alpha and beta activity. Therefore, suitable features should represent the power in the alpha frequency band and the power in the beta frequency band. To accomplish this we use the Logarithmic Band Power technique for the feature extraction. The Logarithmic Band Power is a variation of the Band Power mentioned in Subsection 3.4.2 of Chapter 3. Basically, the logarithm of the Band Power is taken for the Logarithmic Band Power, as the name suggests. Taking the logarithm of the band power ensures that the distribution of the different feature vectors for each epoch of a specific class becomes more normally distributed. For the classifiers we used the Linear Discriminant Analysis machine learning technique, described in Subsection 3.5.1 of Chapter 3, which assumes that the distribution of the feature vectors of each class is normal. Therefore, by making the distribution of the feature vectors closer to a normal distribution, we ensure that the LDA classifiers return more accurate results.

Because we need the Logarithmic Band Power for both the alpha frequency band and the beta frequency band we needed to split the EEG signal in the EEG signal of the alpha frequency band and the EEG signal of the beta frequency band. To extract the EEG signal in the alpha frequency band we applied the Temporal Filter box to use a band pass filter of 8Hz to 12Hz. For the extraction of the EEG signal in the beta frequency the Temporal Filter box was applied to use a band pass filter of 13Hz to 30Hz. The use of the Temporal Filter box was discussed earlier in this subsection. The next steps in the feature extraction using the Logarithmic Band Power are the same for both the alpha frequency band and the beta frequency band. To be able to detect changes over time in the Band Power, we will divide the epochs of each class into smaller epochs of 1 second with an overlap of 50%. The reason for the overlap of 50% lies in the averaging of the powers for the calculation of the Band Power. Without the overlap of 50% of the epochs, there would be no averages taken of the powers in the last 50% of samples from an epoch with the powers in the first 50% of samples in the following epoch. The lack of an average for certain powers of samples that follow each other can be seen as a loss of information. The dividing of the class related epochs in further smaller epochs that overlap can be done in OpenVibe with the Time Based Epoching box. The configuration takes a value in seconds for the duration of the epochs and a value in seconds for the overlap of the epochs. In our case these values were thus 1s and 0.5s. Next, the power of the EEG signal needs to be calculated. In Subsection 3.4.2 we mentioned that the power of a signal is the square of the signal's amplitude. To take the square of the amplitudes in the signal we used the Simple DSP box. The Simple DSP box allows to make calculations on each sample of the incoming EEG signal in OpenVibe by taking one or more signal inputs and providing the result of the calculation to a single output. The formula for the calculation can be entered in the configuration of the Simple DSP box. The next step in calculating the Logarithmic Band Power is to average the powers of each sample in the epoch. This can be done with the Signal Average box. The Signal Average box has no configuration and just simply calculates and outputs the average of each incoming epoch. We now have the Band Power features. To acquire the Logarithmic Band Power features we just need to take the logarithm of the band power. To do this we applied the Simple DSP box with the formula $log(x+1)$. We add 1 to the Band Power to avoid that the logarithm is taken of a Band Power with a value of 0, for which the logarithm is not defined. The last step in the feature extraction is to combine the Logarithmic Band Powers for both the alpha frequency band and the beta frequency band in a feature vector. In OpenVibe a feature vector is created with the Feature Aggregator box that combines the incoming features in each of its inputs into a feature vector.

After the feature extraction comes the *classifier training*. When the feature vector of an epoch for a specific class is constructed it should be accounted for in the trainer of the classifier that will try to detect the class of the feature vector in the online BCI. To train a classifier in OpenVibe we applied the Classifier Trainer box. The Classifier Trainer box starts training a specified machine learning algorithm with the feature vectors it received for each class the classifier should make a distinction between. The training process of the machine learning algorithm starts from the moment all training data is acquired. The end of incoming training data in the EEG signal should therefore be indicated by a trigger label. The Classifier Trainer box

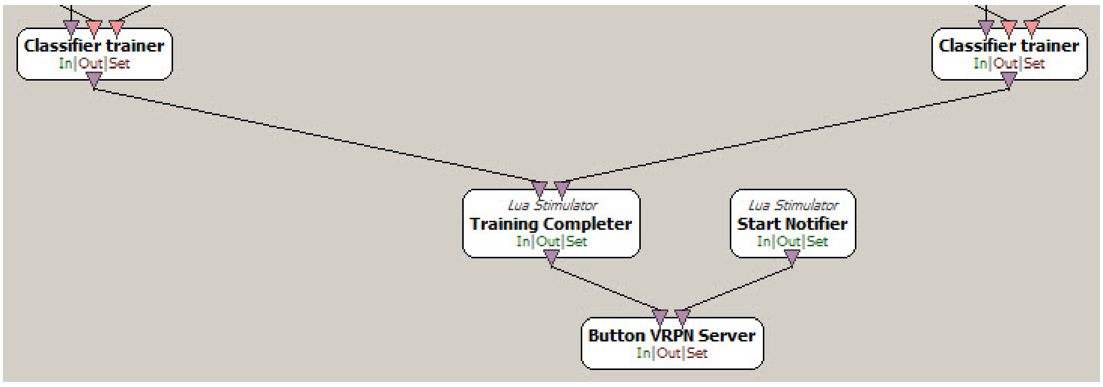


Figure 6.11: End of the BCI training design

will start training the machine learning algorithm when receiving the trigger label and store the configuration for the classifier in a configuration file. When the configuration for the classifier is done the Classifier Trainer will output the label `OVTK_StimulationId_TrainCompleted` to indicated that its done training. The machine learning algorithm for which the training is done, the trigger label and the filename to store the configuration in can be specified in the configuration of the Classifier Trainer box. As mentioned before, we chose the Linear Discriminant Analysis machine learning algorithm, discussed in Subsection 3.5.1. For the trigger label we chose the label `OVTK_GDF_End_Of_Session`. This concludes the training of the classifiers for both Valence and Intensity.
Figure 6.11 shows the last part of the BCI training design. This last part makes sure that the execution of the BCI trainer is terminated correctly. We mentioned in the last paragraph that the Classifier Trainer box outputs the label `OVTK_StimulationId_TrainCompleted` when it is done training. However, we used two classifiers, one for Valence and one for Intensity, which means we used two classifier trainers. This implies that the execution of the BCI trainer should only be terminated when both classifier trainers have finished training. In order to terminate the execution upon the completion of both classifier trainers, we used the Lua Stimulator box. The Lua Stimulator box lets you specify a file in its configuration containing a Lua script. The Lua script can use the values of the label inputs of the Lua Stimulator box, which can be one or more. Based on the label inputs the Lua script can perform some operations and output labels according to these operations. We gave our Lua Stimulator box the suiting name 'Training Completer'. The Lua script we made contains a loop structure that loops until the `OVTK_StimulationId_TrainCompleted` label of both classifier trainers is received. Upon receiving both `OVTK_StimulationId_TrainCompleted` la-

bels the box outputs its own `OVTK_StimulationId_TrainCompleted` label. This output label is then passed on to the VRPN Button Server box. The VRPN Button Server box assigns buttons to each input label. When a specific input label is received, the button related to the label is switched. The switched state is then passed on to VRPN clients. In our case this is our Java component. When the communication layer in our Java component receives the `OVTK_StimulationId_TrainCompleted` label, it automatically sends the label to terminate the OpenVibe BCI trainer through a shared buffer. Besides the Lua Stimulator box to complete the BCI training, we used another Lua Stimulater box that we named 'Start Notifier'. The script of this box automatically sends a label over the VRPN Button Server to indicate that the BCI trainer has started. This way, our Java component is notified that it can begin to show training pictures and label the incoming EEG signal accordingly.

### 6.3.4 BCI Online Design

An overview of the design of the online BCI is given in Figure 6.12. The design itself is less complex than the design of the BCI trainer. The reason for the online BCI design looking simpler than the BCI trainer design is that the feature extraction does not need to happen for each class individually. In the online BCI the incoming EEG signal is not labeled at all, apart from some trigger labels to control the player of the OpenVibe designer, because the labeling of the EEG signal is actual the whole purpose of the online BCI. Furthermore, the preprocessing step and the feature extraction step is roughly the same —apart from some trainer specific steps— as the preprocessing step and feature extraction step of the BCI trainer. It is crucial that both the preprocessing step and the feature extraction are the same in both the BCI trainer and the online BCI in order for the classifiers to interpret features that have the same meaning. Because of the similarity between preprocessing step and the feature extraction step in the BCI trainer design and the online BCI design, we only provide a short overview of the preprocessing step and the feature extraction step in this subsection. For further details on the preprocessing step and feature extraction step we refer to the previous Subsection 6.3.3 discussing the implementation of the BCI trainer.
For the *preprocessing* part we again selected the channels F3 and F4 by applying the Channel Selector box. The next step would be to band pass the EEG signal for the alpha frequency band and the beta frequency band. However, the alpha frequency band and beta frequency band are already band passed in the feature extraction step. Doing this two times would be redundant so we only will band pass the frequency bands in the feature extraction

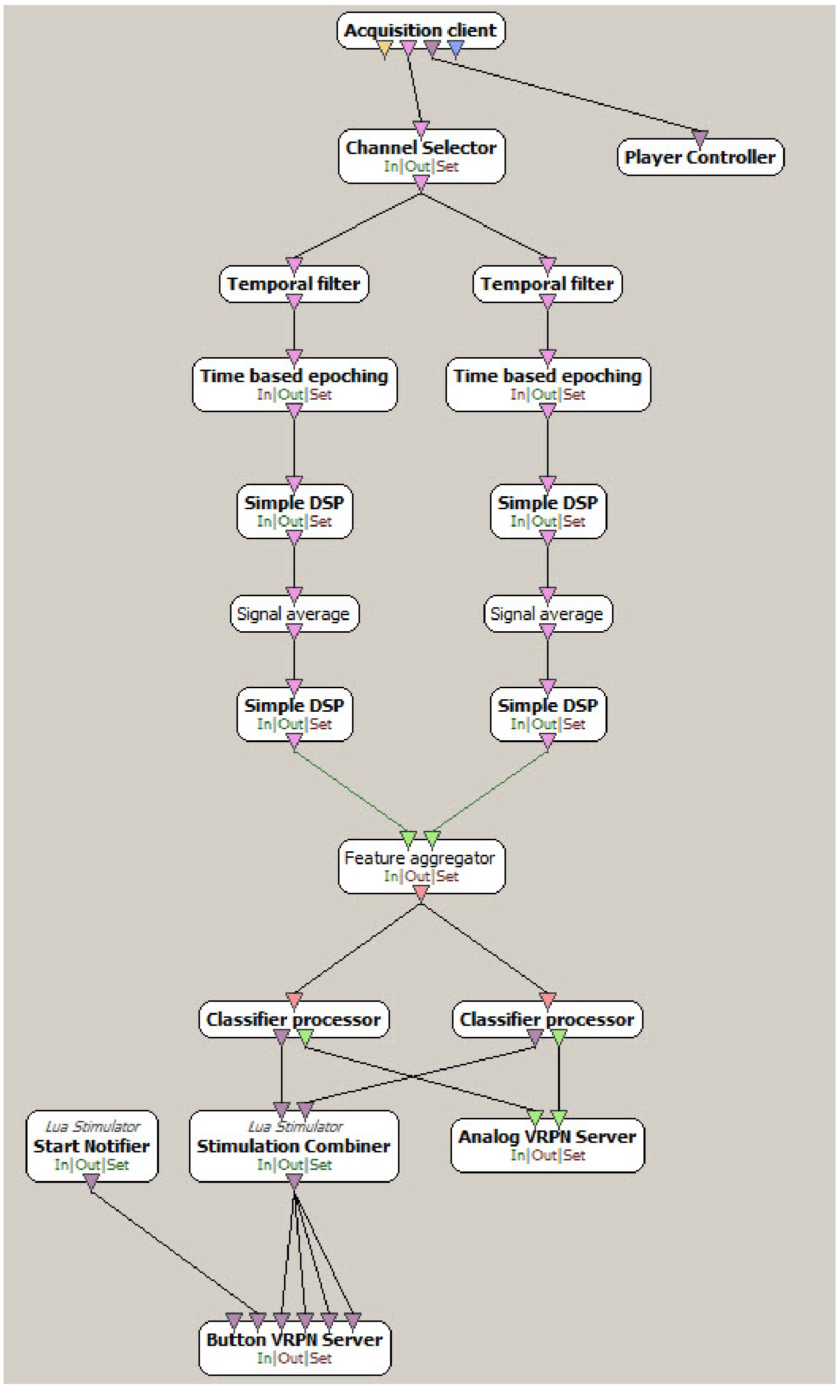


Figure 6.12: Overview of the online BCI design in OpenVibe

step.
For the *feature extraction* step we will again use the Logarithmic Band Power features. Note that for the online BCI we do not epoch the signal according to class labels. Apart from this, the whole calculation is exactly the same as for the BCI trainer. This means that the signal is first split into the EEG signal in the alpha frequency band and the EEG signal in the beta frequency band by using a band pass filter respectively from 8Hz to 12Hz and 13Hz to 30Hz. Next the EEG signal is divided into epochs of 1 second with an overlap of 50% to avoid a certain loss of information in the averages to calculate the Band Power. After epoching the EEG signal for both the alpha frequency band and the beta frequency band, the actual calculation of the Band Power is done. The Band Power is calculated by first calculating the power of the EEG signal. The power of the EEG signal is obtained by taking the square of the amplitude in each sample of each epoch. The next step in calculating the Band Power is to take the average of the power of all the samples in one epoch. The last step is to take the logarithm of the Band Power to obtain the Logarithmic Band Power. The logarithm is used to make the distribution of the feature vectors into a more normal distribution. Linear Discriminant Analysis, which is used as the machine learning algorithm for the classifier, makes the assumption that the incoming feature vectors for each class have a normal distribution. The last step in the feature extraction is to combine the Logarithmic Band Powers over time for both the alpha frequency band power and the beta frequency band power into feature vectors. Finally, in the *classification* step the feature vectors are sent to both the classifier for detecting the difference between negative valence and positive valence, and the classifier for detecting the difference between low arousal and high arousal. Both classifier have two outputs. The first output is the label of the class for which the epoch of the EEG signal has been classified. The second output is, in case of the LDA algorithm, the distance of classified feature vector from the hyperplane. Both classifiers pass on the label of the classified class to a Lua Stimulator box that we named 'Stimulation Combiner'. The Lua script of this box combines the labels of both classifiers into a single label. For instance, the label `OVTK_GDF_Left` from the first classifier and the label `OVTK_GDF_Up` from the second classifier indicate a low valence and a high arousal. In our framework this relates to the emotion 'angry' which is represented by the label `OVTK_StimulationId_Label_03`. The Lua script checks both the input labels and in this case will output the label `OVTK_StimulationId_Label_03`. The Lua Stimulator box will then pass on this label to the Button VRPN Server box which has a button corresponding to each of the four emotion labels. Through changes of a specific button state

received by the VRPN button client, our Java component knows the emotion that was detected. Besides the 'Stimulation Combiner' box, we used another Lua Stimulator box. This box is exactly the same as the 'Start Notifier' Lua Stimulator box of the BCI trainer. This box therefore sends a label to our Java component to indicate that the BCI process has started. Finally, both classifiers pass on the distance to the hyperplane for each classification to an Anolog VRPN Server. Through an analog VRPN client, our Java component can receive these values.

## 6.4 Shared Buffer

To allow external applications to send labels to the OpenVibe Acquisition Server, OpenVibe foresees a C++ header file that creates a buffer that is shared between the external application and the OpenVibe Acquisition Server. Furthermore, the C++ header file includes a method to send labels to the shared buffer that can be picked up by the OpenVibe Acquisition server. To implement this shared buffer the C++ header file uses an instance of message_queue of the Boost C++ libraries[7].
Since we wrote the external application on top of OpenVibe and the header file for the use of the shared buffer provided by OpenVibe is written in C++, we needed to create a Java wrapper for the C++ header file. For creating the Java wrapper the Java Native Interface (JNI)[8] was used. To facilitate the creation of the Java wrapper in JNI we used Simplified Wrapper and Interface Generator (SWIG)[9]. SWIG is a tool that is able to generate wrappers for C++ code to be used by diverse programming and scripting languages[]. One of the programming languages supported by SWIG is Java [6]. To generate the Java wrapper in JNI with SWIG, we created a small project in Visual Studio that compiles into a DLL. We put the C++ header file provided by OpenVibe in the Visual Studio project and we included the Boost libraries that are used by the C++ header file. Furthermore we configured the compiler in Visual Studio to work with SWIG. Upon compiling the Visual Studio project with SWIG, a DLL containing the C++ and JNI code was created as well as the Java code to access the native code in the DLL. However, we adapted the Java code on top of the native code to our own liking by removing unnecessary code, adding error messages and altering some constructors and methods towards simplifying their use. Finally we built a JAR file of

[7] Boost C++ libraries website: `http://www.boost.org/`
[8] Java Native Interface specification: `http://docs.oracle.com/javase/7/docs/technotes/guides/jni/spec/jniTOC.html`
[9] SWIG website: `http://www.swig.org/`

the project containing the Java code on top of the native code to use this JAR file in our Java application on top of OpenVibe.

## 6.5 VRPN

As we already mentioned in Section 6.1, OpenVibe uses a VRPN server to send values and labels to an external application. VRPN stands for Virtual Reality Peripheral Network[10]. As the name might indicate, VRPN is intended to detect state changes of peripheral devices over a network in order to use them virtual reality applications. An example might be the use of a joystick with some buttons and two analog axes, to control an object in a 3D game. The joystick is then calibrated to a VRPN server that detects changes in the On and Off state of the buttons of the joystick and changes in the state of the value of the analog axes of the joystick. The state changes can be obtained by the virtual reality application through a VRPN client that implements a listener.
OpenVibe configures VRPN servers in order for OpenVibe to appear as a device to the VRPN server. OpenVibe can act as an analog device or a button device on a VRPN server. OpenVibe uses an analog VRPN device to send values while a button VRPN device is mainly used by OpenVibe to send labels. To create an analog device on a VRPN server for OpenVibe the box Analog VRPN Server can be used in an OpenVibe scenario. The number of inputs of the Analog VRPN Server box in OpenVibe can be changed and an analog VRPN device with the same number of channels will be configured on a VRPN server. Furthermore, the configuration of the Analog VRPN Server box lets you specify the name that should be used for the analog VRPN device on the VRPN server. An external application can then obtain state changes of the devices by using the device's name. To create a button VRPN device on a VRPN server for OpenVibe, the box Button VRPN Server can be used. For the Button VRPN Server box the number of inputs can be changed as well and a button VRPN device will be created with the same number of buttons. In the configuration of the Button VRPN Server the name for the button VRPN device can be specified as well as the labels to switch a specific button to the on state and to the off state. Because VRPN only makes a distinction between the on and off state of a button of a button device, the labels that correspond to the on and off state of a button should be properly linked in the external application.
To obtain state changes of VRPN devices that are registered to a VRPN server, an application needs to use a VRPN client. The source code of VRPN

[10] VRPN website: `http://www.cs.unc.edu/Research/vrpn/`

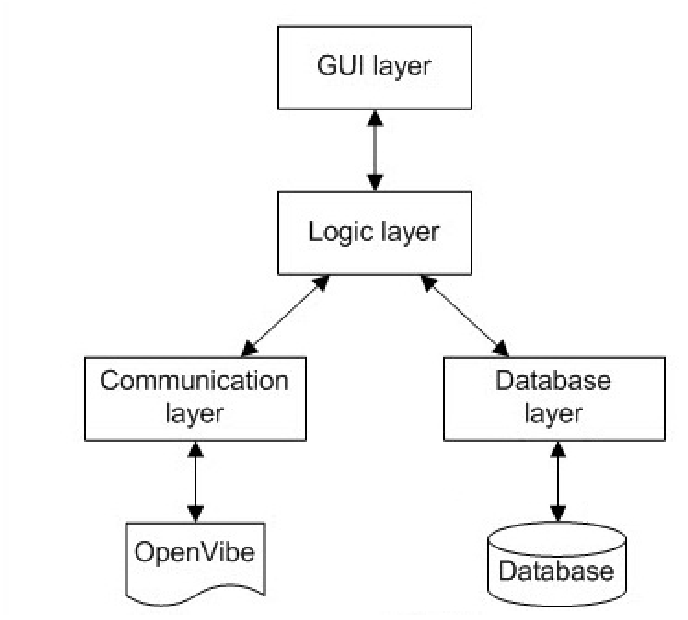


Figure 6.13: Structure of the Java application

is originally written in C++, but they provided a Java wrapper for the VRPN clients. The Java wrapper consists of a DLL containing the original C++ code with JNI code on top, as well as a JAR file that contains Java code that relies on the native code in the DLL. A VRPN client is specific for each type of device for which state changes need to be obtained. To obtain the state changes of a device, a VRPN client uses a listener to detect state changes on the VRPN server. Furthermore, the functionality of handling a state change can be specified in the listener.

## 6.6 Java Application

### 6.6.1 Overview

We implemented the Java application as a layered application. The structure of the Java application can be seen in Figure 6.13. The communication layer handles the communication with OpenVibe. It allows the Java application to send labels to and receive values and labels from OpenVibe, as well as executing BCI designs in OpenVibe. We go further into detail on the communication layer in Subsection 6.6.2. The database layer is used to retrieve images and user information and to store image classification results and user information. More details on the database layer and the database are discussed in Subsection 6.6.3. The logic layer uses both the communication layer and the database layer and is in its turn used by the GUI layer. The logic layer controls the usage of the communication layer and the database layer and links return values of the communication layer to return values of the database layer, which is needed to link labels received from OpenVibe to images retrieved from the database for instance. Additionally the logic layer contains functionality to provide the GUI layer with ready-to-use data. Subsection 6.6.4 discusses the logic layer in more detail. The GUI layer contains the main method from which the application is executed. The GUI layer also contains the screens that make up the user interface and uses the logic layer to control the rest of the application. More details on the GUI layer and its screens are discussed in Subsection 6.6.5.

The communication layer, database layer and logic layer all use a controller that is used to access the functionality of the layer. All calls from layers on top of the layer go through the controller. By using controllers a layer can be used in the implementation of other applications. For instance, suppose an application needs to be build that needs the classification of emotions. The communication, along with the underlying BCI designs in OpenVibe, can directly be used without having to know the actual implementation details of both the communication layer and the OpenVibe BCI designs. Furthermore, a controller uses the implementation of an interface class that contains the actual functionality of the layer. By using implementations of interface classes, the actual implementation of a layer can be easily changed without causing problems for higher layers that use the layer. This allows to use other databases in the database layer or another implementation environment for the BCI designs for instance.

The Java application also contains a package named 'data'. This package contains classes to create objects to transfer data across the layers. For instance, a class is implemented to store an image along with its label of an emotion and the power of the emotion. The data package contains a class with configuration variables as well. All these configuration variables are static finals to allow access to the variables without having to create an instance of the class containing the variables. By combining static final variables in a configuration class we avoid using hard coded configuration. Configuration variables contain path names, names of the training and online BCI designs, label values, and so on.

### 6.6.2 Communication Layer

The communication layer of the Java application is responsible for the communication with the BCI trainer and the online BCI in OpenVibe. Further-

more, the communication layer is also offers the functionality of executing a BCI design in OpenVibe. In Section 6.1 we already mentioned that the Java application sends labels to the OpenVibe Acquisition Server by using a shared buffer and receives labels and data from the OpenVibe BCI design through VRPN servers.
Section 6.4 mentions that we built a Java wrapper for the C++ code to use the shared buffer provide by OpenVibe. To use the shared buffer in the Java application we added the JAR file to use the shared buffer to the build path of our project. We also added the path to the DLL file, containing the native code to use the shared buffer, in the Native Library Location field of the JAR file. The JAR file contains a class named 'StimulationConnection'. By using the constructor of the 'StimulationConnection' class that accepts a String for the name of the shared buffer, a shared buffer with the given name is created. The name to be used for the shared buffer can be changed in the class `Configuration` in the data package, as it isn't hard coded. To send a label to the shared buffer the communication layer calls a method on the `StimulationConnection` class object that accepts the integer value of the label as a parameter.
In Section 6.5 we discussed how VRPN is used by OpenVibe to send values and labels to an external application and that we obtained a Java wrapper to use VRPN client code in our application. To use the VRPN client code we added the VRPN JAR file to the build path of our Java application and specified the path to the DLL in the Native Library Location field of the JAR file. To receive labels and values from the VRPN server created by OpenVibe we implemented a class `LabelReceiver` and `ValueReceiver`. The label receiver uses the VRPN client code from the JAR file for a button device and implements a listener for this button device, while the value receiver uses the VRPN client code from the JAR for an analog device and implements a listener for this analog device. Both receivers use a queue to store state changes. The listeners in the receivers are implemented in such a way that they place incoming state changes in a queue along with the time the state change was received. Additionally, the label receiver links labels to the buttons in order to store the labels into the queue rather than the on state or off state of a button. To instantiate the receivers, the name of the button device and the name of the analog device on the VRPN server for OpenVibe needs to be specified in the constructors of the receivers.
To execute a BCI design in OpenVibe, a simple method in the communication layer can be used. The method accepts a String as a parameter, containing the name of the BCI scenario to be executed. In Subsection 6.3.2 on the OpenVibe Designer we mentioned that a BCI scenario can be executed in

the OpenVibe designer through the command line. The method to execute a BCI scenario in the communication layer basically sends the command to the command line to execute the BCI scenario in the OpenVibe Designer. For this command we needed to use the path to the root folder of the OpenVibe installation. This path is not hard coded and can be specified in the Configuration class of the data package. Furthermore, we also needed the path to the folder containing the OpenVibe BCI scenarios. The path to the OpenVibe BCI scenarios is not hard coded either and can be specified in the Configuration class as well. Finally we added the parameter '–no-gui' to the command which ensures that the BCI is executed in the background. This way the user does not notice any additional application windows popping up when using our application.

### 6.6.3 Database Layer

The database layer handles the retrieval and storage of data with a MySQL database.
The database contains the pictures that will be browsed through in the application by the user as well as a user identification and password for each user. Furthermore, pictures in the database can be linked to a user identification along with the label of an emotion and the valence and arousal of this emotion. The label of an emotion and the valence and arousal of the emotion are linked to a picture and a user from the moment the user browses the picture in the application.
Furthermore, the database contains pictures of the International Affective Picture System (IAPS) which was developed by the National Institute of Mental Health Center for Emotion and Attention at the University of Florida. The IAPS is a collection of pictures that are labeled for the average valence and intensity they evoked in men, women and children[18]. The pictures can be related to emotions by their valence and arousal rating. The pictures in the IAPS were used for training the BCI since the valence and arousal for these pictures is already known, which enables us to label the pictures beforehand.
The database layer retrieves and stores data in the database through JDBC and SQL statements. Layers on top of the database layer are provided with methods to retrieve pictures for the purpose of browsing and labeling, to retrieve pictures for the purpose of training the BCI, store and retrieve user data for the purpose of logging in into the application, and the storage of the labels of emotions and their power linked to the identifier of a picture and the identifier of a user. Pictures from the first database can also be retrieved by emotion through the database layer.

### 6.6.4 Logic Layer

The logic layer lies on top of the communication layer and the database layer, and right under the GUI layer. As the name suggests, the logic layer contains the logic that combines both the communication layer and the database layer in order to provide functionality for the GUI layer. For instance, the logic layer gives access to the GUI layer to methods to start or stop the training process by using the communication layer to let OpenVibe know whether to execute or terminate a BCI process. Furthermore, the logic layer provides a method to the GUI layer to store the classification of a picture in the database based on the id of the picture. The method for storing the classification retrieves the label of the emotion, for which the picture was classified, and the power of the emotion from the communication layer. After retrieving the emotion and the power of the emotion the method passes the id of the picture along with the emotion label and emotion power on to the database layer to store the information. The GUI layer can also obtain pictures through the logic layer for training, browsing or filtered retrieving. The logic layer will keep the time of retrieval of a picture for the purpose of browsing in order to match it to the time of receiving a label. This way the logic layer makes sure that all labels are assigned to the correct picture. Finally, the logic layer also gives access to a method to update the list of images that is to be shown according to some filters for the emotion labels. The method receives the list, the emotion label and the state of the filter through its parameters and updates the list accordingly. Furthermore, the method makes sure that the list is allows sorted by power of the emotion from high to low.

### 6.6.5 GUI Layer

The GUI layer handles the user interface and contains the main method from which the application is executed. In Section 5.2 of Chapter 5 we already discussed what the screens of the Emages application look like with screenshots to illustrate this. In this subsection we discuss some additional details of the GUI based on the implementation.
All screens are created in the GUI layer as JPanels. The JPanel of each screen is added to a CardLayout. Buttons on the screens are used to navigate through the screens by swapping the according JPanel in the CardLayout.
When a user logs in into the Emages application, the GUI layer passes the user credentials on to the logic layer. The logic layer then retrieves the credentials of the user from the database through the database layer and compares them to the credentials received from the GUI layer. The logic layer returns a boolean value to the GUI layer to indicate whether the credentials

match or not in order for the GUI layer to switch to the main screen or not. The GUI layer also calls the logic layer when the training of the BCI needs to be started. The logic layer will then call the communication layer to start the BCI training process in OpenVibe. Also, the logic layer will retrieve training images from IAPS through the database layer and return them to the GUI layer in a list. For the actual slideshow of the training images, we used a thread that is executed from the GUI layer. The thread shows a training image from the list on the screen and sends a call to the logic layer to send the training label of the training image that is shown to the BCI trainer. The logic layer will then pass on the training label to the communication layer which will send it to the BCI trainer in OpenVibe. After the thread shows the training image on the screen and sends the training label to the logic layer, we let it sleep for 5 seconds to make sure that the training image is shown for 5 seconds. When the thread wakes up it changes the training image on the screen with a full black image and we let the thread sleep for 8 seconds. After those 8 seconds the thread wakes up and shows a new training image. This is repeated until all training images in the list are shown. When all the training images are shown, the GUI layer automatically switches the screen to a waiting screen. This screen is shown until the logic layer receives a label from the BCI trainer in OpenVibe through the communication layer. This label indicates that the BCI training process is completed. Eventually, the GUI layer will receive the label from the logic layer and it will switch to the screen to start actual browsing and labeling of images.
The next thing the GUI shows is the browsing screen. The screen in the GUI layer to browse through images looks basically the same as for the BCI training. However, the underlying functionality is different. For the slideshow of the images to browse through, a thread is executed as well from the GUI layer. However, this time the GUI layer will not receive a list of image beforehand. Instead, the thread itself will call the logic layer to retrieve a single image each time a new image needs to be shown. The logic layer will receive the image from the database through the database layer and return it to the slideshow thread of the GUI layer. After the thread sleeps for 5 seconds it wakes up and makes a call to the logic layer. The logic layer will then retrieve the classification result for the shown image of the BCI in OpenVibe through the communication layer. Next, the logic layer passes the id of the image along with the classification result and the id of the user to the database layer in order to store the classification result linked to the image and the user. The thread in the GUI layer then shows a black image and sleeps for 8 seconds, after which it repeats the whole process for the next picture. This continues until the user decides to go back to the main

screen. When leaving the browsing screen, the GUI layer calls the logic layer to shutdown the BCI process to recognise the emotions. The logic layer will pass this message to the communication layer. The communication layer in its turn will send the label to stop to the BCI in OpenVibe.
For retrieving images based on emotions, the GUI provides a screen with filters and a content pane. When the state of a filter is changed, the GUI layer passes on its current list of images as well as the new state of the filter and the related emotion label to the logic layer. If the filter is deselected, the logic layer will remove all the images from the list that are labeled with the emotion of the filter. If the filter is selected, the logic layer will pass on the emotion label to the database layer. The database layer then retrieves all images that are labeled with the emotion label from the database, along with the power of the emotion and the label of the emotion itself. When the logic layer retrieves the list of images, it sorts the list from high to low power of emotion and returns the list to the GUI layer. The GUI layer then updates the content pane.

# 7 Test Setup and Results

## 7.1 Test Setup

To test our application 6 persons between the age of 18 and 26 used the application. 3 of the test participants were male, the other 3 participants were female. The goal of the test experiment is to measure the accuracy of the combined result of the valence and the arousal classifier, as well as the accuracy of both classifiers separately. Furthermore, we want to test whether using pictures that have personalised valence and arousal values for the BCI training has a big influence on the accuracy of the BCI compared to using IAPS pictures.
To start the test experiment we asked each test participant to evaluate 30 pictures based on the valence and arousal evoked by each picture. The evaluation of the pictures was done through a separate application that implemented a self-assessment manikin (SAM) [4]. This application stored the evaluation results of each picture in our database, linked to the username of the participant. Figure 7.1 shows a written version of a SAM form for dimensions valence and arousal. The purpose of the evaluation with the SAM form is to use the most extreme valence and arousal values of the 30 pictures for the BCI training. By using values for valence and arousal that were scored by the participant itself we can create a personalised BCI training.
After evaluating 30 pictures with the SAM, each participant performed two

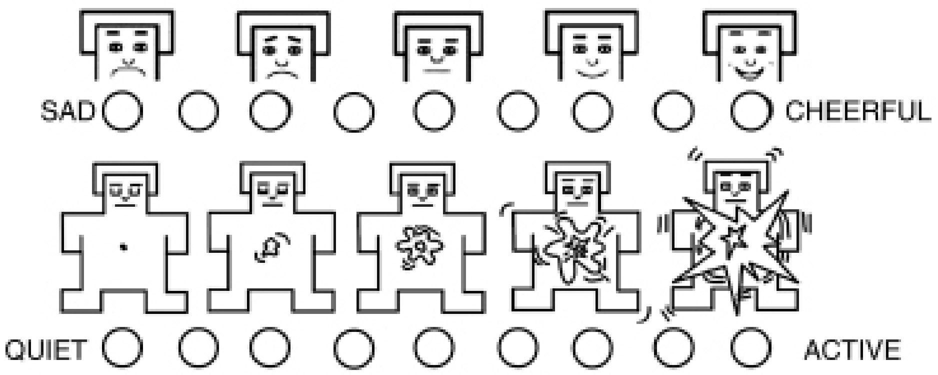


Figure 7.1: Written version of a self-assessment manikin for dimensions valence and arousal (based on [4]

sessions on our Emages application with a rest period in between. The first session used the IAPS pictures and their according valence and arousal values for the BCI training. The second session used the pictures that were evaluated by the participant personally for the BCI training.
At the beginning of each session we placed the Emotiv Epoc EEG capture device on the head of the test participant. Next, we checked the quality of the signal for each electrode of the Emotiv Epoc with the Emotiv Control Panel. When the quality of each electrode was good, we started the OpenVibe Acquisition Server. Each session used 5 pictures for each valence and arousal class for the BCI training process. So 5 pictures were used for high valence, 5 pictures for low valence, etc.
In the first session, which uses IAPS pictures for the BCI training, we showed the black image between each picture for 16 seconds instead of 8 seconds. This was done to give the participant enough time to evaluate each training picture with a written version of the SAM form. Evaluating the IAPS pictures used for training enabled us to detect mismatches between the valence and arousal of the pictures indicated by IAPS and the valence and arousal experienced by the participant. IAPS contains the average valence and arousal values for a large group of people. However, an individual might experience a picture completely different than an average person. This causes the BCI to be trained with wrong EEG data, which could result in a low accuracy.
After completing the BCI training, each test participant browsed through 20 pictures. Finally, we asked each test participant to evaluate the 20 pictures with a written version of the SAM form. The valence and arousal values indicated on the SAM form allowed us to compare the classification results of our BCI to the valence and arousal values experienced by each participant

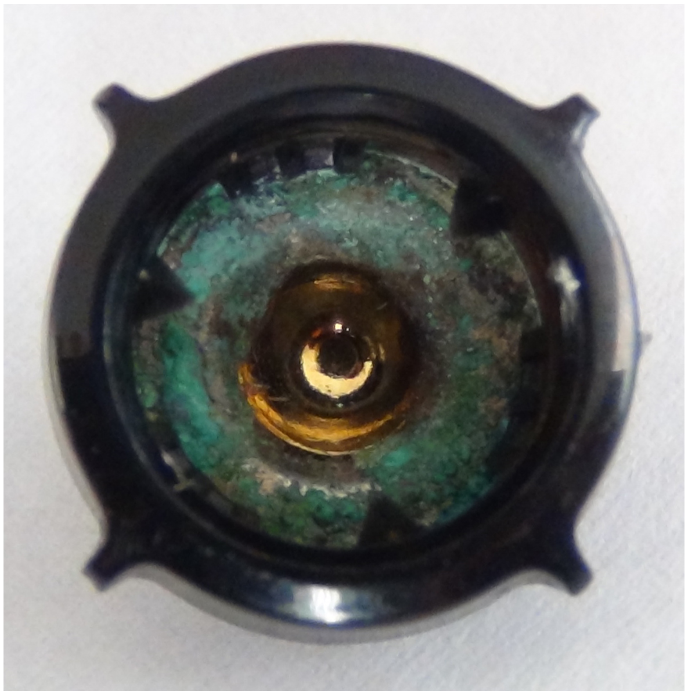


Figure 7.2: Picture taken from an electrode of the Emotiv Epoc we used

personally.
After the first session we held a rest period of 20 minutes before starting the second session. The second session is quite similar to the first session with the exception of using the pictures that were evaluated by each participant personally instead of IAPS pictures. For the training in the second session we only showed the black image in between the training pictures for 8 seconds instead of 16 seconds. The reason for showing the black image for only 8 seconds is that the training pictures did not need to be evaluated with the SAM form anymore as this already happened in the beginning of the test experiment. However, we did still use the SAM form to allow participants to evaluate the 20 browsing pictures.
For both sessions we discarded the test results of a picture, whether they were positive or negative, when distractions in the environment of the test participant occurred. Such distractions are for instance the noise of a passing care or slammed door.

## 7.2 Test Results

In this section we present the results of our test experiment along with a short discussion. A further discussion can be found in Section 8.1 of Chapter 8. Before presenting our test results we would like to mention that the electrodes of the Emotiv Epoc we used had traces of oxidation, as depicted in Figure 7.2. It is not possible to determine the exact impact the state of the electrodes had on the accuracy results we achieved, but it is safe to assume we would have reached higher accuracy results if the electrodes were not oxidised. The accuracy results, as listed in Table 7.1, are obtained by taking the number of classification results that proved to be a correct result and dividing this number by the total number of classification results.
For valence we obtained a mean accuracy of 63.5% with a maximum of 68.3% with the BCI that was trained with pictures from IAPS. With the personalised training pictures we achieved a mean accuracy of 67.9% and a maximum of 77.8%. The IAPS training pictures were correct for 93.3% based upon the corresponding valence values on the SAM forms for each participant.
The classification of arousal has a mean accuracy of 77.8% and a maximum accuracy of 92.0% with training pictures from IAPS. A mean accuracy of 80.4% and a maximum accuracy of 90.0% was achieved with the personalised training pictures. The IAPS training pictures had arousal values that were 83.3% correct.
For the final classification result both a correct valence result and a correct arousal result is needed. Naturally, this means that the accuracy of the total classification result lies lower than the previous accuracies. The mean accuracy of the end result is 46.0% and the maximum accuracy is 57.7% when pictures of IAPS are used for training. With the personalised training pictures we achieve a mean accuracy of 57.1% with a maximum of 66.7%.
Besides calculating the accuracy results, we also calculated the percentages of IAPS training pictures that did not match the valence and arousal values indicated by each test participant on the SAM forms. Mismatches between how a training picture is labeled and how a training picture is actually ex-

| | Valence (Max) | Arousal (Max) | Overall (Max) |
|---|---|---|---|
| IAPS Training | 63.5% (68.3%) | 77.8% (92.0%) | 46.0% (57.7%) |
| Personal Training | 67.9% (77.8%) | 80.4% (90.0%) | 57.1% (66.7%) |

Table 7.1: Accuracy results for training with IAPS pictures and personal pictures

| | Schaaff and Schultz | | Our results | |
|---|---|---|---|---|
| | Approach 1 | Approach 2 | IAPS | Personal |
| Accuracy | 44.0% | 48.9% | 46.0% | 57.1% |
| Max. Accuracy | 47.8% | 66.7% | 57.7% | 66.7% |

Table 7.2: Comparison of our accuracy results to accuracy results of Schaaff and Schultz [36]

perienced emotionally by a user leads to training the classifiers for a class with features that actually belong to an opposite class. As a result, the coefficients of the hyperplanes used by the LDA algorithms in the classifiers are off and can cause inaccurate classification results. For the IAPS pictures used to train the valence classifier, 6.7% of the IAPS valence values indicated the wrong class. As for arousal, 16.7% of the IAPS arousal values indicated a wrong class. As expected, the difference between the accuracy results of with IAPS training and personal training was very low for test participants that indicated a low percentage of mismatches based on the SAM forms they filled in. Furthermore, a bigger difference in accuracy results was observed for test participants that indicate a high mismatch on the SAM forms.
To make a reasonable comparison between the accuracy results we achieved and the results of BCIs in the literature, the same requirements we posed to our BCI need to be fulfilled because these requirements have a direct impact on the accuracy result. The requirements we posed entail that the BCI should be online, e.g. recognise emotions in real-time, and use a limited number of sensors. Furthermore, the emotions to be recognised should be evoked by pictures and both valence and arousal need to be taken into account. In Section 4.2 of Chapter 4 we mentioned the work of Schaaff and Schultz [36], which meets these requirements. Schaaff and Schultz experimented with two BCI designs. The first approach they used resulted in an overall mean accuracy of 44.0% with a maximum 47.8%. With the second approach they achieved a mean accuracy of 48.9% for the end result with a maximum accuracy of 66.7%. The comparison of the accuracy results of Schaaff and Schultz to our accuracy results, as represented in Table 7.2, indicates that we achieved a very acceptable result. Even more so because of the state of the Emotiv Epoc's sensors we used.

# 8 Discussion and Future Work

## 8.1 Discussion

In this thesis we introduced an extensible framework that can be used by applications to recognise a user's emotion in real-time. Our framework offers developers the functionality of using emotion recognition with an EEG-based BCI without requiring a solid background in BCIs and emotion recognition. However, our framework still allows developers, who do have a thorough knowledge of BCIs and emotion recognition, to use their own BCI design by providing an OpenVibe scenario.
We designed the embedded BCI to use as few electrodes as possible while still maintaining an acceptable classification result. The purpose of using as few electrodes as possible is to save the users of applications that are developed with our framework the hassle of setting up complex EEG capturing configurations. A low number of electrodes therefore greatly improves the usability of applications that are built with our framework. Another design choice for our framework was to use an online BCI to provide real-time emotion recognition. Results in real-time greatly expand the possibilities for applications that use our framework.
As a proof of concept we built our own application, that we named Emages, on top of our framework. The Emages application allows users to record their emotions while browsing pictures. However, our framework can be used for

emotion recognition in applications that use any type of content. Through the use of our Emages application we were able to experience application development with our framework first hand. Furthermore, the Emages application allowed to set up a test experiment to measure the accuracy of the BCI embedded in our framework.
We achieved an overall accuracy of 57.1% which is a very acceptable result compared to BCIs in the literature. Our test experiment also pointed out that using training content that was labeled beforehand by users personally resulted in far better accuracy results than accuracy results obtained with a training process that uses pre-labeled content, based on the averages of a group of people, from databases such as IAPS. The accuracy of both valence and arousal separately was only slightly higher—respectively 67.9% versus 63.5% and 80.4% versus 77.8%.—but enough to cause a drastic improvement from 46.0% to 57.1% overall accuracy.
Additionally, the results of our test experiment also pointed out that the classifier for arousal in our BCI achieved a higher accuracy than the classifier for valence. For each of the 6 test participants the accuracy for arousal was higher than or equal to the accuracy for valence in both the BCI trained with IAPS pictures and the BCI trained by personalised pictures. The SAM forms that were filled in by the test participants do not indicate a difference in level between the experienced valence and arousal evoked by the shown pictures. This leads us to conclude that the features extracted from the incoming EEG signal are more meaningful for the classification of arousal than for the classification of valence.
Finally, we like to add that the positions of the electrodes of the Emotiv Epoc greatly vary from person to person due to the way the Emotiv Epoc is designed. For example, the electrodes on the positions F3 and F4 lie further apart for individuals with a larger head and lie extremely close for individuals with a smaller head. Therefore, it is harder to distinguish between signals from the left and right hemisphere of the brain for individuals with smaller heads. The BCI in our framework was designed to differentiate alpha and beta activity in the left hemisphere of the brain from alpha and beta activity in the right hemisphere of the brain. Our test results show a lower arousal, valence and overall accuracy for test participants with smaller heads.

## 8.2 Future Work

The previous section indicated that we were able to reach our goals and obtain a good accuracy result. To reach even better results other preprocessing and feature extraction techniques can be experimented with. In our

discussion we concluded that the extracted features result in a more accurate result for arousal classification than for valence classification. Therefore, experimenting with other feature extractions for the valence classifier would be a good start.
In Section 7.2 of Chapter 7 we mentioned that the electrodes of the Emotiv Epoc we used showed signs of oxidation, which influences accuracy results. Furthermore, we mentioned in our discussion that the position of the electrodes of the Emotiv Epoc greatly varies from person to person. The accuracy of the BCI embedded in our framework greatly depends on distinguishing between left and right brain hemisphere activity. Therefore, it would be advised to test with other EEG capture devices in order to observe if better accuracy is achieved. Also, one of the strong-suits of our framework is that it improves user-friendliness by only using 2 electrodes for measurement and 2 electrodes as reference. In that light a capture device that exploits this advantage can be used or designed by using separate electrodes and an amplifier.
In this thesis we built the Emages application on top of our framework as a proof of concept and to test the accuracy of our framework. Because our framework can be used with other content types besides pictures, applications that use auditory and audio-visual content could be developed on top of our framework in order to further streamline our framework to meet development requirements. Furthermore, building applications on top of our framework that use different types of content enables testing the accuracy of the BCI embedded in our framework for auditory and audio-visual content.

# Bibliography

[1] Athena Akrami, Soroosh Solhjoo, Ali Motie-Nasrabadi, and Mohammad R. Hashemi-Golpayegani. EEG-Based Mental Task Classification: Linear and Nonlinear Classification of Movement Imagery. In *Proceedings of EMBS 2005, 27th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 4626–4629, Shanghai, China, September 2005.

[2] Danny Oude Bos. EEG-based Emotion Recognition The Influence of Visual and Auditory Stimuli, 2006.

[3] Samuel Boudet, Laurent Peyrodie, Philippe Gallois, and Christian Vasseur. A Global Approach for Automatic Artifact Removal for Standard EEG Record. In *Proceedings of EMBS 2006, 28th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 5719–5722, New York, USA, August 2006.

[4] Margaret M. Bradley and Peter J. Lang. Measuring Emotion: The Self-Assessment Manikin and the Semantic Differential. *Journal of Behavior Therapy and Experimental Psychiatry*, 25(1):49–59, 1994.

[5] Neil R. Carlson. *Physiology of Behavior*. Pearson Education, Limited, 2012.

[6] Onur Cinar. *Pro Android C++ with the NDK*. Apress, 2012.

[7] Marco Congedo, Matthieu Goyat, Nicolas Tarrin, Gelu Ionescu, Léo Varnet, Bertrand Rivet, Ronald Phlypo, Nisrine Jrad, Michael Acquadro, and Christian Jutten. "Brain Invaders": A Prototype of an Open-Source P300-based Video Game Working with the OpenViBE Platform. In *Proceedings of BCI 2011, 5th International Brain-Computer Interface Conference 2011*, pages 280–283, Graz, Austria, September 2011.

[8] Roddy Cowie, Ellen Douglas-Cowie, Kostas Karpouzis, George Caridakis, Manolis Wallace, and Spyros Kollias. Recognition of Emotional

States in Natural Human-Computer Interaction. In *Multimodal User Interfaces*, pages 119–153. Springer Berlin Heidelberg, 2008.

[9] Charles Darwin. *The Expression of the Emotions in Man and Animals.* London: John Murray, 1872.

[10] Arnaud Delorme, Terrence Sejnowski, and Scott Makeig. Enhanced Detection of Artifacts in EEG Data using Higher-order Statistics and Independent Component Analysis. *Neuroimage*, 34(4):1443–1449, 2007.

[11] Paul Ekman and Wallace V. Friesen. Constants Across Cultures in the Face and Emotion. *Journal of Personality and Social Psychology*, 17(2):124–129, 1971.

[12] Lawrence A. Farwell and Emanuel Donchin. Talking off the Top of Your Head: Toward a Mental Prosthesis Utilizing Event-Related Brain Potentials. *Electroencephalography and Clinical Neurophysiology*, 70(6):510–523, 1988.

[13] Andrew Jackson, Chet T. Moritz, Jaideep Mavoori, Timothy H. Lucas, and Eberhard E. Fetz. The Neurochip BCI: Towards a Neural Prosthesis for Upper Limb Function. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 14(2):187–190, June 2006.

[14] Tzyy-Ping Jung, Scott Makeig, Colin Humphries, Te-Won Lee, Martin J. Mckeown, Vicente Iragui, and Terrence J. Sejnowski. Removing Electroencephalographic Artifacts by Blind Source Separation. *Psychophysiology*, 37(2):163–178, 2000.

[15] Donald W. Klass. The Continuing Challenge of Artifacts in the EEG. *American Journal of EEG Technology*, 35(4):239–269, 1995.

[16] George H. Klem, Hans O. Lüders, Herbert H. Jasper, and C. Elger. The Ten-Twenty Electrode System of the International Federation. *Electroencephalography and clinical neurophysiology*, 10(2):371–375, 1958.

[17] Wolfgang Klimesch. EEG Alpha and Theta Oscillations Reflect Cognitive and Memory Performance: A Review and Analysis. *Brain Research Reviews*, 29(2-3):169–195, 1999.

[18] Peter J. Lang, Margaret M. Bradley, and B. N. Cuthbert. International Affective Picture System (IAPS): Affective Ratings of Pictures and Instruction Manual. Technical Report A-8, The Center for Research in Psychophysiology, University of Florida, Gainesville, FL, 2008.

[19] Mu Li and Bao-Liang Lu. Emotion Classification Based on Gamma-Band EEG. In *Proceedings of EMBS 2009, 31st Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 1223–1226, Minneapolis, USA, September 2009.

[20] Arthur K. Liu, Anders M. Dale, and John W. Belliveau. Monte Carlo Simulation Studies of EEG and MEG Localization Accuracy. *Human Brain Mapping*, 16(1):47–62, 2002.

[21] Yisi Liu, Olga Sourina, and MinhKhoa Nguyen. Real-Time EEG-Based Emotion Recognition and Its Applications. In *Transactions on Computational Science XII*, pages 256–277. Springer Berlin Heidelberg, 2011.

[22] Scott Makeig, Anthony J. Bell, Tzyy ping Jung, and Terrence J. Sejnowski. Independent Component Analysis of Electroencephalographic Data. In *Proceedings of NIMS 1996, 10th Annual Conference on Advances in Neural Information Processing Systems*, pages 145–151, Denver, USA, December 1996.

[23] Jaakko Malmivuo and Veikko E. Suihko. Effect of Skull Resistivity on the Spatial Resolutions of EEG and MEG. *IEEE Transactions on Biomedical Engineering*, 51(7):1276–1280, 2004.

[24] Albert Mehrabian. Framework for a Comprehensive Description and Measurement of Emotional States. *Genetic, Social & General Psychology Monographs*, 121(3):339–361, 1995.

[25] Gernot R. Müller-Putz and Gert Pfurtscheller. Control of an Electrical Prosthesis with an SSVEP-Based BCI. *IEEE Transactions on Biomedical Engineering*, 55(1):361–364, 2008.

[26] Murugappan Murugappan, Nagarajan Ramachandran, and Yaacob Sazali. Classification of Human Emotion from EEG Using Discrete Wavelet Transform. *Journal of Biomedical Science and Engineering*, 3(4):390–396, 2010.

[27] Dan Nie, Xiao-Wei Wang, Li-Chen Shi, and Bao-Liang Lu. EEG-based Emotion Recognition during Watching Movies. pages 667–670, Cancun, Mexico, April.

[28] Hugh Nolan, Robert Whelan, and Richard B. Reilly. FASTER: Fully Automated Statistical Thresholding for EEG Artifact Rejection. *Journal of Neuroscience Methods*, 192(1):152–162, 2010.

[29] Nordin A.M. Norani, Wan Mansor, and Lee Y. Khuan. A Review of Signal Processing in Brain Computer Interface System, year=2010, month=November, address=Kuala Lumpur, Malaysia, pages=443-449. In *Proceedings of ICBES 2010, IEEE EMBS International Conference on Biomedical Engineering and Sciences.*

[30] Marc R. Nuwer, Giancarlo Comi, Ronald Emerson, Anders Fuglsang-Frederiksen, Jean-Michel Guérit, Hermann Hinrichs, Akio Ikeda, Fransisco J. C. Luccas, and Peter Rappelsburger. IFCN Standards for Digital Recording of Clinical EEG. *Electroencephalography and Clinical Neurophysiology*, 106(3):259–261, 1998.

[31] Ibrahim Omerhodzic, Samir Avdakovic, Amir Nuhanovic, and Kemal Dizdarevic. Energy Distribution of EEG Signals: EEG Signal Wavelet-Neural Network Classifier. *World Academy of Science, Engineering and Technology*, 4(1):1083–1088, 2010.

[32] Robert Plutchik. The Nature of Emotions. *American Scientist*, 89(4):344–350, 2001.

[33] Payam A. Pour, Tauseef Gulrez, Omar Alzoubi, Gaetano Gargiulo, and Rafael A. Calvo. Brain-Computer Interface: Next Generation Thought Controlled Distributed Video Game Development Platform. In *Proceedings of CIG 2008, IEEE Symposium on Computational Intelligence and Games*, pages 251–257, Perth, Australia, December 2008.

[34] James A. Russell. A Circumplex Model of Affect. *Journal of Personality and Social Psychology*, 39(6):1161–1178, 1980.

[35] Saeid Sanei and Jonathon A. Chambers. *EEG Signal Processing.* Wiley-Interscience, 2007.

[36] Kristina Schaaff and Tanja Schultz. Towards Emotion Recognition from Electroencephalographic Signals. In *Proceedings of ACII 2009, 3rd International Conference on Affective Computing and Intelligent Interaction and Workshops*, pages 1–6, Amsterdam, Netherlands, September 2009.

[37] Louis A. Schmidt and Laurel J. Trainor. Frontal Brain Electrical Activity (EEG) Distinguishes Valence and Intensity of Musical Emotions. *Cognition & Emotion*, 15(4):487–500, 2001.

[38] Jesse Sherwood and Reza Derakhshani. On Classifiability of Wavelet Features for EEG-based Brain-computer Interfaces. In *Proceedings of*

*IJCNN 2009, International Joint Conference on Neural Networks*, pages 2508–2515, Atlanta, USA, June 2009.

[39] Ranganatha Sitaram, Andrea Caria, and Niels Birbaumer. Hemodynamic BrainâĂŞComputer Interfaces for Communication and Rehabilitation. *Neural Networks*, 22(9):1320–1328, 2009.

[40] Kenyon Stamps and Yskandar Hamam. Towards Inexpensive BCI Control for Wheelchair Navigation in the Enabled Environment - a Hardware Survey. In *Proceedings of BI 2010, International Conference on Brain Informatics*, pages 336–345, Toronto, ON, Canada, 2010. Springer-Verlag.

[41] Abdulhamit Subasi and Ergun Erçelebi. Classification of EEG signals using Neural Network and Logistic Regression. *Computer Methods and Programs in Biomedicine*, 78(2):87–99, 2005.

[42] Jamie Ward. *The Student's Guide to Cognitive Neuroscience*. Psychology Press, 2010.

[43] Takufumi Yanagisawa, Masayuki Hirata, Youichi Saitoh, Haruhiko Kishima, Kojiro Matsushita, Tetsu Goto, Ryohei Fukuma, Hiroshi Yokoi, Yukiyasu Kamitani, and Toshiki Yoshimine. Electrocorticographic Control of a Prosthetic Arm in Paralyzed Patients. *Annals of Neurology*, 71(3):353–361, 2012.